DEVELOPMENT OF AN ADAPTIVE GRID GENERATION CODE FOR PROJECTILE AERODYNAMICS COMPUTATION

BY

CHYUAN-GEN TU

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN
PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1985

ACKNOWLEDGMENTS

I wish to express my sincere appreciation to my advisor Dr. C. C. Hsu, for his motivation and guidance throughout every stage of this study for the interesting field of adaptive grid generation in projectile aerodynamics computation. I am much obliged to Dr. L. E. Malvern for his help on the writing of this dissertation, and I would like to thank Drs. U. H. Kurzweg, K. Millsaps, and A. K. Varma, for serving on the advisory committee.

I am also deeply indebted to my country for the full support in this study and to Harris Interactive Graphics Laboratory, for the free use of the Harris 800 system in the College of Engineering, University of Florida.

Finally, I wish to express my gratitude to my loving parents and my wife, who have provided me with constant encouragement, support and the inspiration to seek a higher education.

TABLE OF CONTENTS

Abstract of Dissertation Presented to the Graduate School of the University of Florida in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

DEVELOPMENT OF AN ADAPTIVE GRID GENERATION CODE FOR THE PROJECTILE AERODYNAMICS COMPUTATION

BY

CHYUAN-GEN TU

May, 1985

Chairman: C. C. Hsu
Major Department: Engineering Sciences

At the present state-of-technology in the computational aerodynamics, the number of grid points which can be used in solving a complex flow problem is often limited by the capability of existing computer system. Hence, the generation of an optimum grid network is crucial to the accurate solution of a complex flow problem.

A general adaptive grid generation code based on a constrained variational principle is developed by minimizing a general variational functional which consists of functionals for measuring the smoothness, orthogonality, and concentration of grid. The coefficients (penalty parameter) of the functionals of orthogonality and concentration can be determined by dimensional analysis; moreover, they can be treated as variables to enhance the corresponding property locally. The one-dimensional variational principle has been readily applied to adaptive boundary grid generation as well

as two-dimensional case in the domain, and excellent consistancy is achieved with the same corresponding penalty parameters. The developed one-dimensional grid generator can be applied to any complex configuration of boundary such as curvilinear surface combined with polygonal shape.

The adaptive grid generation technique developed with the artificial clustering is successfully applied to the computation of projectile inviscid transonic flow problem. An excellent agreement has been obtained between the pressure computed on the adaptive grid network and experimental surface pressure. Also, this agreement is better than the agreement obtained from the solution computed on the fixed grid network which is generated by "GRIDGEN."

CHAPTER I
INTRODUCTION

In recent years, the study of boundary-fitted grid systems has become important and active because of the versatile applications of these systems in engineering problems. These applications have made the finite-difference method a very effective tool for solving complex fluid dynamics problems. In general, a grid system in the physical space is numerically generated either by solving a set of partial differential equations with specified conditions or by other methods such as algebraic interpolation from predetermined boundary points and/or along chosen grid lines. For the boundary-fitted grid system, a general curvilinear coordinate is developed, such that all boundaries of the arbitrary two-dimensional region of interest coincide with coordinate curves. Thus, finite difference expressions at the boundary may be applied using exact grid points on the intersections of curvilinear coordinate lines without the need for any interpolation between points of the grid. The avoiding of interpolation is particularly important for boundaries with strong curvature or slope discontinuities, both of which are common in physical applications. Likewise, interpolation between grid points not coincident with the boundaries is especially inaccurate with a governing

system of equations that produces large gradients in the vicinity of the boundaries, and the character of solution may be significantly altered in such cases [1]. The generation of a boundary-fitted curvilinear coordinate system which needs to employ the technique of coordinate transformation between physical space and computational space is, therefore, an important part of a numerical algorithm for highly accurate solution.

A good grid system generated for fluid flow computations can be justified from the smoothness of grid distribution, the minimal skewness of grids and the grid resolution being adaptive to solution in the physical space. It has been shown that rapid changes of grid size and highly skewed grids can result in undesirable errors [2]. Also, it is well known in the approximate solution that the choice of high concentration of grid in regions where the solution gradient is very large is crucial to the accuracy of the numerical result. In fact, an improper grid resolution in high gradient regions can be detrimental to the solution accuracy as well as to the convergence process of a computational algorithm for fluid flow problems [3]. Therefore, the success of a finite difference method in the computational space for solving complex fluid dynamics problems can depend strongly upon the generation of a good adaptive grid system in the physical space.

In the last few years, many different adaptive grid generation methods have been proposed and reported in a

symposium [4], a workshop [5], and a conference [6]. It seems that each proposed method has its own advantages for the special problem treated, but it may not be suitable for other complex problems. However, the method based on a constrained variational principle, proposed by Brackbill [7], is more general and shows great promise for complex problems in which physical phenomena such as shock waves, flames, and viscous layers may cause extremely large gradient regions of unknown orientation. In this method, the governing equations of the adaptive grid generation are derived by minimizing a general variational functional. This functional consists of a functional for measuring the smoothness of grid and two constrained functionals, one for the orthogonality of grids and the other for the concentration of grids. The relative importance of the two penalty parameters on adaptive grid generation had been investigated for simple problems and reported in [7], and these two parameters were treated as constants to date. Saltzman and Brackbill [8,9] have applied the method to a two-dimensional inviscid supersonic flow past a step in a wind tunnel with rectilinear boundaries; they obtained striking results with an adaptive grid generator that caused the computational grid to move with the shock fronts. The grid points on the rectilinear boundary are obtained by extrapolating from the corresponding interior grid lines in a simple linear relation so that the interior grid lines will be orthogonal to the boundary. This method may be cumbersome for the

curvilinear boundaries or more complicated body shapes such as projectiles.

A better design of flight vehicles, aerodynamic devices, and sheltering structures can depend strongly upon an accurate prediction of aerodynamic forces. There are quite a few effective finite-difference schemes and solution algorithms developed for solving complete and/or modified Navier-Stokes equations; the choice of a good grid system for the approximation may, however, be crucial to the success of these routines. For a complex flow problem, an extremely large number of grids is often required to attain satisfactory results; consequently, an optimum or proper adaptive gridding becomes essential for accurate aerodynamic force computations with limited computer resources.

Steger [10], at NASA Ames Research Center, developed a thin-layer Navier-Stokes code about arbitrary two-dimensional geometrics. Later, Pulliam and Steger [11] extended the thin-layer code to the three-dimensional case. A combination of general curvilinear coordinate transformations and an implicit approximate factorization technique which was developed by Warming and Beam [12] was employed in that code; so that the fine grid sizes required for spatial accuracy and viscous resolution do not impose stringent stability limitations. The present state of the technology in computational aerodynamics clearly indicates that the thin-layer code can provide accurate solutions for high speed aerodynamics problem [11,13]. Also, Nietubicz,

Pulliam, and Steger [14] have successfully applied the thin-layer Navier-Stokes code to solve axisymmetric flow problems. They modified the code by use of a cylindrical coordinate system, and the three-dimensional governing equations are simplified for flow fields which are invariant in the circumferential direction. Moreover, this axisymmetric thin-layer code has readily been applied to transonic projectile aerodynamics problems at the U.S. Army Ballistic Research Laboratory [15]. Their report shows that good agreement has been obtained between the computed and experimented surface pressure. However, the successful application of the code depends strongly upon a good adaptive grid network generated from a grid generation code named GRIDGEN [16]. For the procedure of grid generation in GRIDGEN, the boundary grid points are generated by the clustering which is implemented with a cubic function that allows the user to specify the first and the last grid point spacing along the grid line of each segment of the boundary. A planar grid network within the domain can be generated by solving either an elliptic boundary value problem or a hyperbolic initial value problem in the current version of GRIDGEN. The resulting grid network can then be modified to provide high grid resolution in the viscous layer by clustering grid points along the grid lines normal to the streamwise direction. Finally, a three-dimensional grid network for a projectile aerodynamic problem is comprised of a sequence of the generated planar grid networks around the axis of the

projectile. As reported by Sturck [17], an accurate solution for the aerodynamics of an axisymmetric transonic projectile can be obtained with a good adaptive grid network. This grid network is generated by GRIDGEN only after considerable experimentation with boundary grid positioning and grid density distribution.

For a transonic projectile at an angle of attack, the three-dimensional grid network generated by GRIDGEN may not be a proper one for use in the computation, since the flow field is not axisymmetric and consequently a planar grid network adaptive to the flow field at an azimuthal location may not be a good grid network for use at other azimuthal planes. Moreover, for projectiles at supersonic speed and other high speed viscous flow problems such as blast aerodynamics [18], the regions where the solution gradients are very large are not known in advance; consequently, a cut and try approach for finding a good grid network can be very tedious. Therefore, it is extremely important and essential for solving complex flow problems to develop effective techniques for systematic generation of good adaptive grid networks.

The prime purpose of this study is to investigate and develop a two-dimensional adaptive grid generation technique for high speed projectile aerodynamics computation. The secant-ogive-cylinder-boattail projectile with sting and the axisymmetric version of the thin-layer Navier-Stokes code of the U.S. Army Ballistic Research Laboratory are employed in

this development. The adaptive grid system is mainly based on a constrained variational principle; hence, the governing equations for the transformation are quasi-linear and must be solved numerically by a stable iterative method. The detailed investigation is performed in this study for several different solution methods. The method of Newton-Raphson iteration with successive over relaxation, which has been shown to be a highly efficient and accurate method in solving nonlinear boundary value problems, is employed here to solve the set of complicated partial differential equations. The investigations conducted on the constant penalty parameters for orthogonality and concentration of grid points have shown that these two parameters can be treated as variables over the field by using dimensional analysis to enhance the properties of the control function obtained from the approximate solutions. The adaptive boundary grid generation developed by applying a one-dimensional variational principle provides complete consistency between the grid points within the domain and corresponding points on the boundary. The developed adaptive grid generation code coupled with an axisymmetric thin-layer Navier-Stokes code has demonstrated the need for clustering the grid points in the vicinity of the body surface to obtain good results. This coupled code gives computational results which agree well with the experimental data.

CHAPTER II
ADAPTIVE GRID GENERATION

II.1 Two-Dimensional Coordinate Transformation

For a two-dimensional coordinate system, the general transformation can be an one-to-one mapping of an irregular boundary curve surface in the physical plane (x,z) onto a rectangular boundary in the computational plane (ξ,ζ), as shown in Fig. 2.1. The solution in the transformed computational plane is then given by the functions

$$\begin{aligned} \xi &= \xi(x,z) \\ \zeta &= \zeta(x,z) \end{aligned} \tag{2.1}$$

The grid of the computational plane is a square mesh with equal grid spacing in a fixed rectangular field with MxN grid points. The fluid flow problems are then solved on this computational plane. Similarly, the inverse transformation is given by

$$\begin{aligned} x &= x(\xi,\zeta) \\ z &= z(\xi,\zeta) \end{aligned} \tag{2.2}$$

where ξ, ζ are the natural coordinates. The interior grid points in the physical plane are the intersection points of

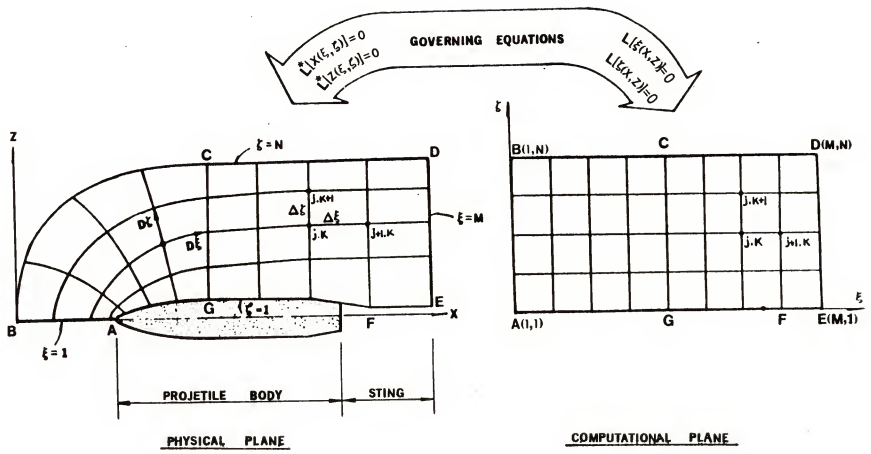


Fig. 2.1 Coordinate transformation

the natural coordinate curves ξ = const. and ζ = const., which intersect the boundary in the chosen boundary grid points. This forms the boundary-fitted grid system. Thus, the integer values j and k of the natural coordinates (ξ,ζ) define a point (j,k) in the computational plane, and the mapping associates with it a point $[x_{j,k},z_{j,k}]$ in the physical plane. In formulating the grid generation problem, mathematically the grid $[x_{j,k},z_{j,k}]$ is viewed as the image of a mapping $[x(\xi,\zeta),z(\xi,\zeta)]$ in which only the points corresponding to the integer values of the natural coordinates, (ξ,ζ), are realized. Conversely, the image of a computation grid of curvilinear quadrilateral cells in the physical plane is a uniform, rectilinear grid in the (ξ,ζ) plane with spacing defined as

$$\Delta\xi = \Delta\zeta = 1 \tag{2.3}$$

A grid generator determines the mapping $[x(\xi,\zeta),z(\xi,\zeta)]$.

The system of governing equations for the mapping can be defined by numerical transformation procedures including techniques based on elliptic and hyperbolic partial differential equations, algebraic or geometric procedures based on parametric surfaces or interpolation techniques, and analytic transformation procedures such as conformal mapping. To generate an adaptive grid system, in this work we use partial differential equations as the governing equations for the transformation of the physical plane to

the computational plane. We then define

$$L[\xi(x,z)] = 0$$
$$L[\zeta(x,z)] = 0 \tag{2.4}$$

where L[ ] is an operator; thus Eqn. (2.4) represents some type of partial differential equations (PDE) with Dirichlet boundary conditions such that $\zeta = 1$, $\zeta = N$, $\xi = 1$, and $\xi = M$ of the computational plane are on the corresponding segments AE, BD, AB, and DE of the physical plane (see Fig. 2.1). Since it is desired to perform all numerical computations in the uniform rectangular transformed plane, the dependent and independent variables in Eqn. (2.4) must be interchanged. This yields the coupled system of equations in the L*[ ] operator form

$$L^*[x(\xi,\zeta)] = 0$$
$$L^*[z(\xi,\zeta)] = 0 \tag{2.5}$$

These equations are subject to the transformed boundary conditions which are suitably chosen functions that map the known shape of four segments AE, AB, BE, and DE of the physical boundaries into the corresponding segments AE ($\zeta = 1$), AB ($\xi = 1$), BD ($\zeta = N$), and DE ($\xi = M$) on the rectilinear boundaries of the chosen computational domain.

Usually, the system given by the equations of (2.5) is considerably more complex than that defined by Eqn. (2.4);

however, the boundary conditions of Eqn. (2.5) are specified on straight boundaries and the coordinate spacing in the computational plane is uniform.

II.2 Review of Variational Principles

A variational principle specifies a scalar functional, I*, which is defined by an integral form

$$I^* = \int_{\Omega} A(U)d\Omega + \int_{\Gamma} B(U)d\Gamma \tag{2.6}$$

where A and B are the operators such that A(U) and B(U) represent the constraint equations in terms of the unknown vector function U (i.e., U = {U}), in a domain Ω, and on the boundary Γ. The solution to the continuum problem is a function U which makes I* stationary with respect to small changes δU, or in other words, the function U which minimizes I* is in some sense the "best" choice. Thus, for a solution to the continuum problem, the variation is

$$\delta I^* = 0$$

If a variational principle can be found, then immediately means are established for obtaining approximate solutions in the standard, integral form suitable for adaptive grid generation analysis.

A variational principle may also be considered as the problem of making a functional I* stationary, subject to the unknown vector U obeying some set of additional differential constraint equations

$$C(U) = 0 \quad \text{in } \Omega$$
$$E(U) = 0 \quad \text{on } \Gamma \tag{2.7}$$

where C and E are the operators such that $C(U) = \{C(U)\}$ and $E(U) = \{E(U)\}$. We can introduce these constraint equations by forming another functional

$$I = I^* + \lambda_\Omega \int_\Omega C^T(U)C(U)d\Omega + \lambda_\Gamma \int_\Gamma E^T(U)E(U)d\Gamma \tag{2.8}$$

in which λ_Ω and λ_Γ are the penalty parameters or so-called "penalty numbers" in reference [19], and superscript "T" stands for transpose of the matrix. The products, C^TC and E^TE, are

$$C^TC = [C_1^2, C_2^2, \ldots]$$

$$E^TE = [E_1^2, E_2^2, \ldots]$$

and the quantities of $C^T(U)C(U)$ and $E^T(U)E(U)$ must always be positive. Substituting Eqn. (2.6) into Eqn. (2.8) gives

$$I = I_\Omega + I_\Gamma \tag{2.9}$$

where I_Ω is the functional within the domain Ω,

$$I_\Omega = \int_\Omega A(U)d\Omega + \lambda_\Omega \int_\Omega C^T(U)C(U)d\Omega, \tag{2.10}$$

and I_Γ is the functional on the boundary Γ,

$$I_\Gamma = \int_\Gamma B(U)d\Gamma + \lambda_\Gamma \int_\Gamma E^T(U)E(U)d\Gamma, \tag{2.11}$$

II.3 Governing Equations

Since ξ and ζ are curvilinear coordinates in the physical plane as shown in Fig. (2.1), the differential vectors

$$\{D\xi\} = \begin{Bmatrix} \frac{\partial x}{\partial \xi} \\ \\ \frac{\partial z}{\partial \xi} \end{Bmatrix} d\xi \quad ; \quad \{D\zeta\} = \begin{Bmatrix} \frac{\partial x}{\partial \zeta} \\ \\ \frac{\partial z}{\partial \zeta} \end{Bmatrix} d\zeta \tag{2.12}$$

can be defined from the relationship between the Cartesian and curvilinear coordinates, and they are directed tangentially to the ζ = const. and ξ = const. curves, respectively. A useful observation is that the differential properties of the mapping determine the properties of the computations grid. For example, the magnitude of the vector $\{D\xi\}$ can be related to grid spacing Δs by taking the finite difference form of the Eqn. (2.12) with $\Delta\xi \equiv \Delta\zeta \equiv 1$,

$$\begin{aligned} |D\xi|_{j,k} &= \left[\left(\frac{\partial x}{\partial \xi}\right)^2 + \left(\frac{\partial z}{\partial \xi}\right)^2\right]^{1/2} d\xi_{j,k} \\ &\approx [(\Delta x)^2 + (\Delta z)^2]^{1/2}_{j,k} = \Delta s \end{aligned} \tag{2.13}$$

where Δs is the grid spacing in the physical plane between two points (j,k) and (j+1,k). Similarly, since the length

of the vector resulting from a cross product of $\{D\xi\}\cdot\{D\zeta\}$ is related to the area of the elementary parallelgram in the curvilinear coordinate system, we obtain

$$d(\text{Area}) = J^{-1} d\xi d\zeta \ , \tag{2.14A}$$

where J^{-1} is the Jacobian determinant, defined as

$$J^{-1} = \frac{\partial(x,z)}{\partial(\xi,\zeta)} = \begin{vmatrix} x_\xi & x_\zeta \\ z_\xi & z_\zeta \end{vmatrix} = x_\xi z_\zeta - z_\xi x_\zeta \tag{2.15}$$

Thus, the finite difference form of Eqn. (2.14A) becomes

$$\Delta(\text{Area}) = J^{-1}\Delta\xi\Delta\zeta = J^{-1} \tag{2.14B}$$

$$= \text{grid area in physical space}$$

As described in Chapter I, a good grid system generated for fluid flow computations can be justified from the smoothness of grid distribution, the orthogonality of grids and the grid resolution being adaptive to a control weight function obtained from the solution in the physical space. Thus, the governing equations of the adaptive grid generation are derived from minimizing a general variational functional as defined in Eqn. (2.10)

$$I_\Omega = I_s + \lambda_o I_o + \lambda_w I_w \tag{2.16}$$

The functional for measuring the smoothness of grids is I_s which can be written as

$$I_s = \int_\Omega [(\nabla\xi\cdot\nabla\xi) + (\nabla\zeta\cdot\nabla\zeta)]dxdz \qquad (2.17)$$

Thus, minimizing this integral results in Laplace's equations for ξ and ζ as functions of x and z. Because of the well-known averaging properties of solutions of Laplace's equation, we might expect a grid constructed in this way to be, in some sense, smooth [20].

The functional for measuring the orthogonality of grids is I_o

$$I_o = \int_\Omega (\nabla\xi\cdot\nabla\zeta)^2(\frac{1}{J})^3 dxdz \ , \qquad (2.18)$$

in which $(\nabla\xi\cdot\nabla\zeta)$ is zero when the conjugate curves (ξ = const., ζ = const.) of the grid are completely orthogonal. Also, the inclusion of the $(\frac{1}{J})^3$, the cubic of the Jacobian, weights the measure in favor of orthogonalizing regions of large cell area [9]; this weighting function can also reduce problems with rounding errors [7].

The functional for measuring the concentration of grids is I_w

$$I_w = \int_\Omega WJ^{-1} dxdz \qquad (2.19)$$

where $W = W(x,z)$ is a given weighting function and J^{-1} represents the grid area in the curvilinear coordinate

system as derived in Eqn. (2.14B). If we were to minimize this integral, we would predict that where W is large, J^{-1} (cell area) should be small and conversely where J^{-1} is large, W should be relatively small. The parameters, λ_o and λ_w, are specified coefficients whose values give more or less emphasis to orthogonality or to concentration, respectively. The procedure for optimizing the grid is to choose a mapping that minimizes the functional I_Ω of Eqn. (2.16).

It is useful to normalize the functionals, I_o and I_w, in order to keep the order of magnitude of each functional the same. Thus, three normalization factors, λ_o', λ_w', and λ are defined by

$$\lambda_o \equiv \frac{\lambda}{\lambda_o'} \quad ; \quad \lambda_w \equiv \frac{\lambda}{\lambda_w'} \tag{2.20}$$

in which λ is the specified global constant parameter. A fairly effective and simple way to determine λ_o' and λ_w' is to apply a dimensional analysis. For instance, the derivatives $\frac{\partial x}{\partial \xi}, \frac{\partial z}{\partial \zeta}$, . . . , etc. are proportional to L/L', where L is the physical length and L' is the parameter length (points number). The order of magnitude of the Jacobian can be written as

$$J^{-1} = \frac{\partial x}{\partial \xi}\frac{\partial z}{\partial \zeta} - \frac{\partial z}{\partial \xi}\frac{\partial x}{\partial \zeta} \sim \left(\frac{L}{L'}\right)^2 \tag{2.21}$$

The integrand of the smoothness functional in Eqn. (2.17) has the order of $(L'/L)^2$ and, similarly, the integrands of the orthogonality and concentration functionals are $(L/L')^2$, $\bar{W}(L/L')^2$, respectively, where $\bar{W}$ is the unit of measure of the weight function within the domain. Thus, the order of magnitude of the functionals is

$$I_s \sim (\frac{L'}{L})^2 L^2 = (L')^2 \tag{2.22}$$

$$I_o \sim (\frac{L}{L'})^2 L^2 \tag{2.23}$$

$$I_w \sim \bar{W}(\frac{L}{L'})^2 L^2$$

In order that I_o and I_w have the same order of magnitude as I_s, λ_o' and λ_w' should have

$$\lambda_o' \sim (\frac{L}{L'})^4 \tag{2.24}$$

$$\lambda_w' \sim \bar{W}(\frac{L}{L'})^4$$

Specific choices for L, L', and $\bar{W}$ were made by Saltzman and Brackbill [8]; L was the maximum length in the physical domain, L' was the number of points in one direction, and $\bar{W}$ was the area average of W(x,z), such that $\bar{W} = \frac{1}{\Omega} \int_\Omega W d\Omega$. Thus, Eqn. (2.16) becomes

$$I_\Omega = I_s + \frac{\lambda}{\lambda_o'} I_o + \frac{\lambda}{\lambda_w'} I_w$$

$$= \int_\Omega (\nabla\xi\cdot\nabla\xi + \nabla\zeta\cdot\nabla\zeta)d\Omega + \frac{\lambda}{\lambda_o'}\int_\Omega (\nabla\xi\cdot\nabla\zeta)^2(\frac{1}{J})^3 d\Omega$$

$$+ \frac{\lambda}{\lambda_w'}\int_\Omega WJ^{-1}d\Omega \qquad (2.25)$$

Next, the independent and dependent variables are interchanged by the relations

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} x_x & x_z \\ z_x & z_z \end{bmatrix} = \begin{bmatrix} x_\xi & x_\zeta \\ z_\xi & z_\zeta \end{bmatrix}\begin{bmatrix} \xi_x & \xi_z \\ \zeta_x & \zeta_z \end{bmatrix} \qquad (2.26)$$

where $\xi_x = \frac{\partial\xi}{\partial x}$, $\zeta_x = \frac{\partial\zeta}{\partial x}$, . . . , etc., are the metrics. Thus, they have the following relations

$$\begin{bmatrix} \xi_x & \xi_z \\ \zeta_x & \zeta_z \end{bmatrix} = \begin{bmatrix} x_\xi & x_\zeta \\ z_\xi & z_\zeta \end{bmatrix}^{-1} \equiv J\begin{bmatrix} z_\zeta & -x_\zeta \\ -z_\xi & x_\xi \end{bmatrix} \qquad (2.27)$$

where J is the Jacobian of transformation such that $J = 1/J^{-1}$ in which J^{-1} is defined in Eqn. (2.15). The functionals of Eqn. (2.25) can be readily derived by using the relations of Eqn. (2.27); the variational integral becomes

$$I_{\Omega'} = \int_{\Omega'} J(\nabla x \cdot \nabla x + \nabla z \cdot \nabla z) d\Omega' +$$

$$\frac{\lambda}{\lambda_o'} \int_{\Omega'} (x_\xi x_\zeta + z_\xi z_\zeta)^2 d\Omega' +$$

$$\frac{\lambda}{\lambda_w'} \int_{\Omega'} W(x,z)(\frac{1}{J})^2 d\Omega' \qquad (2.28)$$

where $d\Omega' = d\xi d\zeta$ is the area element in the computational domain. The necessary conditions for $x(\xi,\zeta)$ and $z(\xi,\zeta)$ to minimize $I_{\Omega'}$ give the Euler equations

$$0 = [\frac{\partial}{\partial x} - \frac{\partial}{\partial \xi}\frac{\partial}{\partial x_\xi} - \frac{\partial}{\partial \zeta}\frac{\partial}{\partial x_\zeta}][J(\nabla x \cdot \nabla x + \nabla z \cdot \nabla z) +$$

$$\frac{\lambda}{\lambda_o'} (x_\xi x_\zeta + z_\xi z_\zeta)^2 + \frac{\lambda}{\lambda_w'} (\frac{W}{J^2})]$$

$$0 = [\frac{\partial}{\partial z} - \frac{\partial}{\partial \xi}\frac{\partial}{\partial z_\xi} - \frac{\partial}{\partial \zeta}\frac{\partial}{\partial z_\zeta}][J(\nabla x \cdot \nabla x + \nabla z \cdot \nabla z) +$$

$$\frac{\lambda}{\lambda_o'} (x_\xi x_\zeta + z_\xi z_\zeta)^2 + \frac{\lambda}{\lambda_w'} (\frac{W}{J^2})]$$

Performing the above differentiations and collecting coefficents of the highest derivative terms result in the coupled quasi-linear elliptic system of equations

$$a_1x_{\xi\xi}+a_2x_{\xi\zeta}+a_3x_{\zeta\zeta}+b_1z_{\xi\xi}+b_2z_{\xi\zeta}+b_3z_{\zeta\zeta} = -(\frac{\lambda}{\lambda'_w})\frac{W_x}{2J^2}$$

$$b_1x_{\xi\xi}+b_2x_{\xi\zeta}+b_3x_{\zeta\zeta}+c_1z_{\xi\xi}+c_2z_{\xi\zeta}+c_3z_{\zeta\zeta} = -(\frac{\lambda}{\lambda'_w})\frac{W_x}{2J^2} \qquad (2.29)$$

where $W_x = W_\xi\xi_x + W_\zeta\zeta_x$ and $W_z = W_\xi\xi_z + W_\zeta\zeta_z$ can be transformed by applying Eqn. (2.27) to the computational form

$$W_x = J(z_\zeta W_\xi - z_\xi W_\zeta)$$

$$W_z = J(x_\xi W_\zeta - x_\zeta W_\xi) \qquad (2.30$$

The coefficients in Eqn. (2.29) are given by

$$\begin{Bmatrix} a_i \\ b_i \\ c_i \end{Bmatrix} = \begin{bmatrix} a_{si} & a_{oi} & a_{wi} \\ b_{si} & b_{oi} & b_{wi} \\ c_{si} & c_{oi} & c_{wi} \end{bmatrix} \begin{Bmatrix} 1 \\ \lambda/\lambda'_o \\ \lambda/\lambda'_w \end{Bmatrix}, \quad i = 1,2,3 \qquad (2.31)$$

in which the corresponding coefficients of smoothness, orthogonality and concentration are

$$\begin{bmatrix} a_{s1} & a_{s2} & a_{s3} \\ b_{s1} & b_{s2} & b_{s3} \\ c_{s1} & c_{s2} & c_{s3} \end{bmatrix} = \begin{bmatrix} \alpha A^* & -2\beta A^* & \gamma A^* \\ -\alpha B^* & 2\beta B^* & -\gamma B^* \\ \alpha C^* & -2\beta C^* & \gamma C^* \end{bmatrix}, \qquad (2.32)$$

$$\begin{bmatrix} a_{o1} & a_{o2} & a_{o3} \\ b_{o1} & b_{o2} & b_{o3} \\ c_{o1} & c_{o2} & c_{o3} \end{bmatrix} = \begin{bmatrix} x_\zeta^2 & 2(2x_\xi x_\zeta + z_\xi z_\zeta) & x_\xi^2 \\ x_\zeta z_\zeta & x_\xi z_\zeta + x_\zeta z_\xi & x_\xi z_\xi \\ z_\zeta^2 & 2(x_\xi x_\zeta + 2z_\xi z_\zeta) & z_\xi^2 \end{bmatrix} , \quad (2.33)$$

$$\begin{bmatrix} a_{w1} & a_{w2} & a_{w3} \\ b_{w1} & b_{w2} & b_{w3} \\ c_{w1} & c_{w2} & c_{w3} \end{bmatrix} = W \begin{bmatrix} z_\zeta^2 & -2z_\xi z_\zeta & z_\xi^2 \\ -x_\zeta z_\zeta & x_\xi z_\zeta + x_\zeta z_\xi & -x_\xi z_\xi \\ x_\zeta^2 & -2x_\xi x_\zeta & x_\xi^2 \end{bmatrix} , \quad (2.34)$$

respectively, and

$$A^* = z_\xi^2 + z_\zeta^2 , \; B^* = x_\xi z_\xi + x_\zeta z_\zeta , \; C^* = x_\xi^2 + x_\zeta^2$$

$$\alpha = (x_\zeta^2 + z_\zeta^2)/J^3 , \quad \beta = (x_\xi x_\zeta + z_\xi z_\zeta)/J^3 , \quad (2.35)$$

$$\gamma = (x_\xi^2 + z_\xi^2)/J^3$$

For convenience, Eqn. (2.29) may also be written in the operator form,

$$LA[x] + LB[z] = -\lambda_w \left(\frac{W_x}{2}\right)\left(\frac{1}{J}\right)^2$$

$$LB[x] + LC[z] = -\lambda_w \left(\frac{W_z}{2}\right)\left(\frac{1}{J}\right)^2 \quad (2.36)$$

where $\lambda_w \equiv \lambda/\lambda'_w$, and

$$LA \equiv a_1 \frac{\partial}{\partial \xi^2} + a_2 \frac{\partial}{\partial \xi \partial \zeta} + a_3 \frac{\partial}{\partial \zeta^2}$$

$$LB \equiv b_1 \frac{\partial}{\partial \xi^2} + b_2 \frac{\partial}{\partial \xi \partial \zeta} + b_3 \frac{\partial}{\partial \zeta^2} \tag{2.37}$$

$$LC \equiv c_1 \frac{\partial}{\partial \xi^2} + c_2 \frac{\partial}{\partial \xi \partial \zeta} + c_3 \frac{\partial}{\partial \zeta^2}$$

II.4 Boundary Conditions

The consistency between boundary and domain is important for the success of the adaptive grid generation. Thus, it is necessary to generate the adaptive grids on the boundary. Saltzman and Brackbill [8] have applied the method based on a variational principle to a two-dimensional inviscid supersonic flow past a step in a wind tunnel with rectilinear boundaries and obtained striking results. Boundary grids are extrapolated from interior grids of the domain orthogonal to the boundary. However, this is difficult to apply to extrapolating on a curvilinear boundary such as a projectile with curved ogive or some other irregular body configuration. Therefore, a one-dimensional variational procedure must be applied to properly determine the adaptive boundary grid, as follows.

Similarly to the coordinate transformation as described in sec. II.3, let S be a variable along a curved boundary in physical space. Then, S is mapped

correspondingly to a variable ξ on a line ζ = const. in the computational space as shown in Fig. 2.2, and conversely, such that

$$\begin{aligned} \xi &= \xi(S) = j \\ S &= S(\xi) = S_j \\ j &= 1,2, \ldots, M \end{aligned} \tag{2.38}$$

The first derivative S_ξ can be considered a measure of the grid spacing ΔS by the mean value theorem, since $\Delta\xi = 1$.

$$\Delta S = S_j - S_{j-1} = \frac{dS(S^*)}{d\xi} \ , \ S_{j-1} < S^* < S_j \tag{2.39}$$

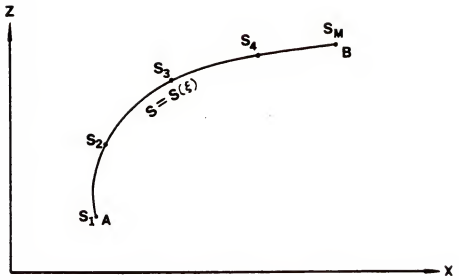
Thus S_ξ represents the grid spacing in one-dimensional space, analogous to J^{-1}, the cell area in two-dimensional space for $\Delta\xi = \Delta\zeta = 1$.

To optimize the physical characteristics of grid smoothness and concentration on the boundary, we seek to minimize their functionals in accordance with the one-dimensional variation principles. Thus, the functional of Eqn. (2.11), which measures the properties of the mapping, becomes

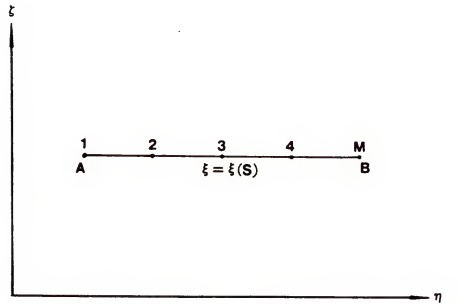
$$I_\Gamma = I_S + \lambda_w I_W \tag{2.40}$$

where I_S, the functional of smoothness, is

$$I_S = \int_{S_1}^{S_M} (\xi_S)^2 \, dS \tag{2.41}$$



Physical space



Computational space

Fig. 2.2 1-D Coordinate transformation

and I_w, the functional of concentration, is

$$I_w = \int_{S_1}^{S_M} W(S) S_\xi \, dS \tag{2.42}$$

in which W(S) is a specified weight function which makes the boundary grid spacing small corresponding to the large value of W(S). The parameter λ_w also becomes $\lambda_w \equiv \lambda/\lambda'_w$ as defined and determined in the two-dimensional case by the dimensional analysis.

The order of magnitude for the functionals of smoothness and concentration are

$$I_s \sim (\frac{L'}{L})^2 L \tag{2.43}$$

$$I_w \sim \bar{W}(\frac{L}{L'})L \tag{2.44}$$

where L, a total length of boundary, can be determined by $L = \int_\Gamma d\Gamma$, L' is the number of corresponding points, and $\bar{W}$ is the length average of W(S), such that $\bar{W} = \frac{1}{L}\int_\Gamma W d\Gamma$. Thus, λ'_w is determined by matching the order of I_s to I_w

$$\lambda'_w = \bar{W}(\frac{L}{L'})^3 \tag{2.45}$$

After interchange of the independent and dependent variables, Eqn. (2.40) becomes

$$I(\xi) = \int_1^M (S_\xi)^{-1}\, d\xi + \frac{\lambda}{\lambda'_W} \int_1^M W\cdot(S_\xi)^2\, d\xi \qquad (2.46)$$

The necessary condition for $S(\xi)$ to minimize $I(\xi)$ gives Euler's equation

$$0 = [\frac{\partial}{\partial S} - \frac{\partial}{\partial \xi}\frac{\partial}{\partial S_\xi}][S_\xi^{-1} + \frac{\lambda}{\lambda'_W}(WS_\xi^2)] \qquad (2.47)$$

Collecting coefficients of the highest derivative yields the equation,

$$S_{\xi\xi} = -(\frac{\lambda}{\lambda'_W})\, W_S S_\xi^5/[2(1 + \frac{\lambda}{\lambda'_W} S_\xi^3 W)] \qquad (2.48)$$

where $W_S = W_\xi/S_\xi$, and the end points are $\xi(S_1) = 1$, $\xi(S_M) = M$. Notice that λ may be the same as that used in Eqn. (2.29), because of the requirement of consistency between boundary and domain.

II.5 Methods of Solution

In order to generate a grid network, the governing equations, Eqn. (2.36), are solved by the finite difference method. The values of the derivatives at the nodes are approximated by the second-order central differences. For the use of an alternating direction implicit method, Eqn. (2.36) can be approximated in either the ξ- or ζ-direction, such that for the k^{th} line (row):

$$a_1 x_{j-1,k} - 2(a_1+a_3)x_{j,k} + a_1 x_{j+1,k} + b_1 z_{j-1,k} - 2(b_1+b_2)z_{j,k}$$
$$+ b_1 z_{j+1,k} = (S_x)_{j,k}$$
$$b_1 x_{j-1,k} - 2(b_1+b_3)x_{j,k} + b_1 x_{j+1,k} + c_1 z_{j-1,k} - 2(c_1+c_2)z_{j,k} \tag{2.49}$$
$$+ c_1 z_{j+1,k} = (S_z)_{j,k}$$

where

$$(S_x)_{j,k} = -\frac{\lambda}{\lambda_w'}\left[\frac{1}{2}\left(\frac{1}{J}\right)^2 W_x\right]_{j,k} - \left[a_2 x_{\xi\zeta} + b_2 z_{\xi\zeta}\right]_{j,k}$$
$$- a_3\left[x_{j,k-1} + x_{j,k+1}\right] - b_3\left[z_{j,k-1} + z_{j,k+1}\right],$$

$$(S_z)_{j,k} = -\frac{\lambda}{\lambda_w'}\left[\frac{1}{2}\left(\frac{1}{J}\right)^2 W_z\right]_{j,k} - \left[b_2 x_{\xi\zeta} + c_2 z_{\xi\zeta}\right]_{j,k}$$
$$- b_3\left[x_{j,k-1} + x_{j,k+1}\right] - c_3\left[z_{j,k-1} + z_{j,k+1}\right],$$

and for the j^{th} line (column):

$$a_3 x_{j,k-1} - 2(a_1+a_3)x_{j,k} + a_3 x_{j,k+1} + b_3 z_{j,k-1} - 2(b_1+b_3)z_{j,k}$$
$$+ b_3 z_{j,k+1} = (S_x')_{j,k}$$
$$b_3 x_{j,k-1} - 2(b_1+b_3)x_{j,k} + b_3 x_{j,k+1} + c_3 z_{j,k-1} - 2(c_1+c_3)z_{j,k} \tag{2.50}$$
$$+ c_3 z_{j,k+1} = (S_z')_{j,k}$$

where

$$(S'_x)_{j,k} = -\frac{\lambda}{\lambda'_w}[\frac{1}{2}(\frac{1}{J})^2 W_x]_{j,k} - [a_2 x_{\xi\zeta} + b_2 z_{\xi\zeta}]_{j,k}$$

$$- a_1[x_{j-1,k} + x_{j+1,k}] - b_1[z_{j-1,k} + z_{j+1,k}],$$

$$(S'_z)_{j,k} = -\frac{\lambda}{\lambda'_w}[\frac{1}{2}(\frac{1}{J})^2 W_z]_{j,k} - [b_2 x_{\xi\zeta} + c_2 z_{\xi\zeta}]_{j,k}$$

$$- b_1[x_{j-1,k} + x_{j+1,k}] - c_1[z_{j-1,k} + z_{j+1,k}],$$

in which W_x and W_z, the derivatives of the weight function are also approximated by the 2nd order central difference,

$$(W_x)_{j,k} = [J(z_\zeta W_\xi - z_\xi W_\zeta)]_{j,k}$$

$$= J_{j,k}[(z_{j,k+1} - z_{j,k-1})(W_{j+1,k} - W_{j-1,k}) -$$

$$(z_{j+1,k} - z_{j-1,k})(W_{j,k+1} - W_{j,k-1})]/4$$

$$(W_z)_{j,k} = [J(x_\xi W_\zeta - x_\zeta W_\xi)]_{j,k}$$

$$= J_{j,k}[(x_{j+1,k} - x_{j-1,k})(W_{j,k+1} - W_{j,k-1}) -$$

$$(x_{j,k+1} - x_{j,k-1})(W_{j+1,k} - W_{j-1,k})]/4$$

Also, the coefficients of Eqn. (2.31) which are in terms of the derivatives are approximated by the central difference form.

There are many methods to solve the Eqn. (2.49) or Eqn. (2.50). Since the governing equations are a coupled quasilinear set of equations, the finite difference

approximations are solved by iteration. In order to obtain an effective calculation for solving the grid system equation, we investigate several methods to identify which one is the best. Methods of Newton-Raphson iteration, implicit line iteration, and alternating direction implicit are the quite general and efficient solvers. These methods continually iterate with successive overrelaxation (SOR) until the residual errors of the solutions are everywhere less than the allowed error.

II.5.1 Newton-Raphson Iteration (N-R)

Let Rx and Rz be the residual errors of the Eqn. (2.36), that is,

$$(Rx)_{j,k} = \{LA[x] + LB[z] + (\frac{\lambda}{\lambda'_w})\frac{W_x}{2}(\frac{1}{J})^2\}_{j,k}$$

$$(Rz)_{j,k} = \{LB[x] + LC[z] + (\frac{\lambda}{\lambda'_w})\frac{W_z}{2}(\frac{1}{J})^2\}_{j,k} \qquad (2.51)$$

In the iteration process, the values of $x_{j,k}$ and $z_{j,k}$ are calculated point by point from Eqn. (2.51), such that first we consider a point $(x^{\ell},z^{\ell})_{j,k}$ which is not a root of equations of $(Rx)^{\ell}_{j,k} = 0$ and $(Rz)^{\ell}_{j,k} = 0$, but is reasonably close to the roots, where the superscript ℓ is the iteration number. We expand $Rx^{\ell+1}$ and $Rz^{\ell+1}$ in a Taylor series about $x^{\ell}_{j,k}$ and $z^{\ell}_{j,k}$, and retain only the linear terms.

$$(Rx)^{\ell+1}_{j,k} \approx (Rx)^{\ell}_{j,k}+(x^{\ell+1}-x^{\ell})_{j,k}\left[\frac{\partial(Rx)_{j,k}}{\partial x_{j,k}}\right]^{\ell} + (z^{\ell+1}-z^{\ell})_{j,k}\left[\frac{\partial(Rx)_{j,k}}{\partial z_{j,k}}\right]^{\ell}$$

$$(Rz)^{\ell+1}_{j,k} \approx (Rz)^{\ell}_{j,k}+(x^{\ell+1}-x^{\ell})_{j,k}\left[\frac{\partial(Rz)_{j,k}}{\partial x_{j,k}}\right]^{\ell} + (z^{\ell+1}-z^{\ell})_{j,k}\left[\frac{\partial(Rz)_{j,k}}{\partial z_{j,k}}\right]^{\ell} \qquad (2.52)$$

where $(Rx)^{\ell}_{j,k} = Rx(x^{\ell}_{j,k},z^{\ell}_{j,k})$ and $(Rz)^{\ell}_{j,k} = Rz(x^{\ell}_{j,k},z^{\ell}_{j,k})$, in which j = 2, . . ., M-1 and k = 2, . . ., N-1. An approximate value of the root $(x^{\ell+1},z^{\ell+1})_{j,k}$ can be obtained by setting $Rx^{\ell+1} = Rz^{\ell+1} = 0$ at the left-hand side of (2.52) to yield

$$0 = (Rx)^{\ell}_{j,k}+(x^{\ell+1}-x^{\ell})_{j,k}\left[\frac{\partial(Rx)_{j,k}}{\partial x_{j,k}}\right]^{\ell} + (z^{\ell+1}-z^{\ell})_{j,k}\left[\frac{\partial(Rx)_{j,k}}{\partial z_{j,k}}\right]^{\ell}$$

$$0 = (Rz)^{\ell}_{j,k}+(x^{\ell+1}-x^{\ell})_{j,k}\left[\frac{\partial(Rz)_{j,k}}{\partial x_{j,k}}\right]^{\ell} + (z^{\ell+1}-z^{\ell})_{j,k}\left[\frac{\partial(Rz)_{j,k}}{\partial z_{j,k}}\right]^{\ell} \qquad (2.53)$$

where the four derivative terms can be readily found by differentiating the finite difference form of Eqn. (2.49) or

(2.50) but note that there are no contributions when differentiating their source terms, $(S_x)_{j,k}$, $(S_z)_{j,k}$ or $(S'_x)_{j,k}$, $(S'_z)_{j,k}$ (i.e., $\partial[(S_x)_{j,k}]/\partial x_{j,k} = \partial[(S_x)_{j,k}]/\partial z_{j,k} = \ldots = 0$). Thus, there results

$$\frac{\partial(Rx)_{j,k}}{\partial x_{j,k}} = -2(a_1+a_3)_{j,k}$$

$$\frac{\partial(Rx)_{j,k}}{\partial z_{j,k}} = \frac{\partial(Rz)_{j,k}}{\partial x_{j,k}} = -2(b_1+b_3)_{j,k} \qquad (2.54)$$

$$\frac{\partial(Rz)_{j,k}}{\partial z_{j,k}} = -2(c_1+c_3)_{j,k}$$

Solving for $x^{\ell+1}_{j,k}$ and $z^{\ell+1}_{j,k}$ from Eqn. (2.53) gives

$$x^{\ell+1}_{j,k} = x^{\ell}_{j,k} + \frac{1}{D_{j,k}}\left[(Rz)_{j,k}\frac{\partial(Rx)_{j,k}}{\partial z_{j,k}} - (Rx)_{j,k}\frac{\partial(Rz)_{j,k}}{\partial z_{j,k}}\right]^{\ell}$$

$$z^{\ell+1}_{j,k} = z^{\ell}_{j,k} + \frac{1}{D_{j,k}}\left[(Rx)_{j,k}\frac{\partial(Rz)_{j,k}}{\partial x_{j,k}} - (Rz)_{j,k}\frac{\partial(Rx)_{j,k}}{\partial x_{j,k}}\right]^{\ell} \qquad (2.55)$$

where

$$D_{j,k} = \left[\frac{\partial(Rx)_{j,k}}{\partial x_{j,k}} \cdot \frac{\partial(Rz)_{j,k}}{\partial z_{j,k}} - \frac{\partial(Rx)_{j,k}}{\partial z_{j,k}} \cdot \frac{\partial(Rz)_{j,k}}{\partial x_{j,k}}\right]^{\ell}$$

For the SOR iterations, $x^{\ell+1}_{j,k}$ and $z^{\ell+1}_{j,k}$ are calculated point by point as

$$x_{j,k}^{\ell+1} = x_{j,k}^{\ell} + \omega[(x_{j,k}^{*})^{\ell+1} - x_{j,k}^{\ell}]$$
$$z_{j,k}^{\ell+1} = z_{j,k}^{\ell} + \omega[(z_{j,k}^{*})^{\ell+1} - z_{j,k}^{\ell}] \qquad (2.57)$$

where $x_{j,k}^{*}$ and $z_{j,k}^{*}$ are the updated values obtained on the current iteration of Eqn. (2.55). The quantity ω is a pure number in the range $0 < \omega < 2$, which is called the relaxation factor.

The application to the one-dimensional case is similar. Thus, the residual error of Eqn. (2.48) is defined as

$$(Rs)_j = \{S_{\xi\xi} + (\frac{\lambda}{\lambda'_w}) W_s S_\xi^5 / [2(1 + \frac{\lambda}{\lambda'_w} S_\xi^3 W)]\}_j , \qquad (2.59)$$

$j = 2, \ldots, M-1$

The Taylor series expansion about S_j^{ℓ} gives

$$(Rs)_j^{\ell+1} \approx (Rs)_j^{\ell} + (S_j^{\ell+1} - S_j^{\ell})[\frac{\partial (Rs)_j^{\ell}}{\partial S_j}] \qquad (2.60)$$

where $(Rs)_j^{\ell} = Rs(S_j^{\ell})$ and the derivative term, $\frac{\partial (Rs)_j}{\partial S_j}$, is simply calculated by differentiating the discretized central difference form of Eqn. (2.59),

$$\partial [(Rs)_j]^{\ell} / \partial S_j = -2, \qquad (2.61)$$

Setting $(Rs)_j^{\ell+1} = 0$ in (2.60) and solving for $S_j^{\ell+1}$ point by point give

$$S_j^{\ell+1} = S_j^{\ell} + \Delta S_j^{\ell} \tag{2.62}$$

where

$$\Delta S_j^{\ell} = \frac{1}{2}(Rs)_j^{\ell} \tag{2.63}$$

Finally, the N-R method with the SOR procedure added into Eqn. (2.62) is formed as

$$S_j^{\ell+1} = S_j^{\ell} + \omega[(S^*)_j^{\ell+1} - S_j^{\ell}] \tag{2.64}$$

in which $(S^*)_j^{\ell+1}$ is the updated value obtained on the current iteration of Eqn. (2.62). The factor ω may be specified as the same quantity as used in Eqn. (2.57).

This method can be implemented by an alternating direction such that

Iterations = Odd number	Iterations = Even number
k = 2, 3, ... , N-1	j = 2, 3, ... , M-1
Eqn. (2.57)	Eqn. (2.57)
j = 2, 3, ... , M-1	k = 2, 3, ... , N-1

II.5.2 Implicit Line Iteration (I.L.)

This method is to apply a block iterative method implemented line by line. In order to do this, the left hand side of Eqn. (2.49) needs to be assembled into block

matrix form. For example, in Eqn. (2.49), let us define $A = -2(a_1+a_3)$, $B = -2(b_1+b_3)$, and $C = -2(c_1+c_3)$, also set $M = N = 6$ such that $j = 1, 2, \ldots, 6$; $k = 1, 2, \ldots, 6$. The block matrix is readily formed in the following block tridiagonal form for the k^{th} row:

$$\left[\begin{array}{cc|cc|cc|cc} A & B & a_1 & b_1 & & & & \\ B & C & b_1 & c_1 & & & & \\ \hline a_1 & b_1 & A & B & a_1 & b_1 & & \\ b_1 & c_1 & B & C & b_1 & c_1 & & \\ \hline & & a_1 & b_1 & A & B & a_1 & b_1 \\ & & b_1 & c_1 & B & C & b_1 & c_1 \\ \hline & & & & a_1 & b_1 & A & B \\ & & & & b_1 & c_1 & B & C \end{array}\right]^{\ell} \left\{\begin{array}{c} x_{2,k} \\ z_{2,k} \\ \hline x_{3,k} \\ z_{3,k} \\ \hline x_{4,k} \\ z_{4,k} \\ \hline x_{5,k} \\ z_{5,k} \end{array}\right\}^{\ell+1}$$

$$= \left\{\begin{array}{l} (S_x)_{2,k} - a_1 x_{1,k} - b_1 z_{1,k} \\ (S_z)_{2,k} - b_1 x_{1,k} - c_1 z_{1,k} \\ \hline (S_x)_{3,k} \\ (S_z)_{3,k} \\ \hline (S_x)_{4,k} \\ (S_z)_{4,k} \\ \hline (S_x)_{5,k} - a_1 x_{6,k} - b_1 z_{6,k} \\ (S_z)_{5,k} - b_1 x_{6,k} - c_1 z_{6,k} \end{array}\right\}^{\ell} \qquad (2.65)$$

$k = 2, \ldots, N-1$

The right hand side of Eqn. (2.65) contains known values from the previous iteration "ℓ" and the boundary conditions. Computational algorithm implemented row by row, and the unknown values are solved by linearlizing their coefficients in the block matrices. Similarly, for j^{th} column, Eqn. (2.50) can also be formed and executed column by column to obtain the solutions, such that the form is

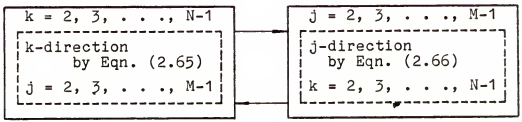
$$\begin{bmatrix} A & B & a_3 & b_3 & & & & \\ B & C & b_3 & c_3 & & & & \\ \hline a_3 & b_3 & A & B & a_3 & b_3 & & \\ b_3 & c_3 & B & C & b_3 & c_3 & & \\ \hline & & a_3 & b_3 & A & B & a_3 & b_3 \\ & & b_3 & c_3 & B & C & b_3 & c_3 \\ \hline & & & & a_3 & b_3 & A & B \\ & & & & b_3 & c_3 & B & C \end{bmatrix}^{\ell} \begin{Bmatrix} x_{j,2} \\ z_{j,2} \\ \hline x_{j,3} \\ z_{j,3} \\ \hline x_{j,4} \\ z_{j,4} \\ \hline x_{j,5} \\ z_{j,5} \end{Bmatrix}^{\ell+1}$$

$$= \begin{Bmatrix} (S'_x)_{j,2} - a_3 x_{j,1} - b_3 z_{j,1} \\ (S'_z)_{j,2} - b_3 x_{j,1} - c_3 z_{j,1} \\ \hline (S'_x)_{j,3} \\ (S'_z)_{j,3} \\ \hline (S'_x)_{j,4} \\ (S'_z)_{j,4} \\ \hline (S'_x)_{j,5} - a_3 x_{j,6} - b_3 z_{j,6} \\ (S'_z)_{j,5} - b_3 x_{j,6} - c_3 z_{j,6} \end{Bmatrix}^{\ell} \qquad (2.66)$$

$j = 2, \ldots, M-1.$

II.5.3 Alternating Direction Implicit (ADI)

We combine Eqn. (2.65) and (2.66) by implementing a first iteration for k^{th} line in the row direction and then a second iteration for the j^{th} line in the column direction for a complete cycle of iteration. Such methods are aptly designated "alternating direction implicit" (ADI) methods. The process of ADI is briefly described as follows:



II.6 Numerical Experiments and Results

The Harris 800 computer of the engineering college of the University of Florida was employed to perform the computations. The numerical experiments were performed to find an optimal relaxation factor ω. These experiments also examine the effects of orthogonality and concentration, and analyze the efficiency and accuracy of the results. Two experimental problems have the same size domain with rectilinear boundaries and 12x12 grid points (i.e., $M = N = 12$). However, the distribution of the grid points on the boundaries is different; for the first problem, the grid spacing is an arbitrary choice; for the second, the grid

spacing is uniform. In the physical space, the ranges of x and z are specified by

$$x : [x_1, x_M] = [1,12]$$
$$z : [z_1, z_N] = [1,12] \tag{2.67}$$

The allowable error, R_o, is defined as the error of allowable residual. For convenience, the finite difference form of residual Eqn. (2.51) is rewritten here as

$$(Rx)_{j,k} = [LA(x)+LB(z)+\lambda_w \frac{W_x}{2}(\frac{1}{J})^2]_{j,k}$$
$$(Rz)_{j,k} = [LB(x)+LC(z)+\lambda_w \frac{W_z}{2}(\frac{1}{J})^2]_{j,k} \tag{2.68}$$

where j = 2, 3, . . ., M-1; k = 2, 3, . . ., N-1. If every point (j,k) satisfies the conditions

$$(Rx)_{j,k} \leq R_o$$
$$(Rz)_{j,k} \leq R_o$$

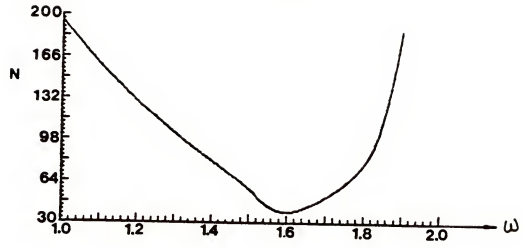
then we find an approximate solution with this specified allowable error R_o. The number of iterations and the amount of CPU time required to reach an approximate solution satisfying these conditions at every grid point are determined by the numerical experiment.

Since the three methods are implemented with SOR, it is desirable to find an optimal relaxation factor, in order to

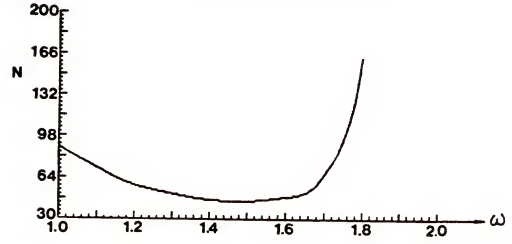
obtain the highest efficiency for each method. The first problem is investigated for this purpose. The results are shown in Fig. 2.3, which illustrates the relation between the number of iterations (N) and the relaxation factor (ω). These results are obtained with $\lambda_o = \lambda_w = 1$ and the maximum allowable error, $R_o = 10^{-6}$, for every point within the domain. These results obtained by the N-R method in Fig. 2.3(a) and by I.L. method in Fig. 2.3(c) clearly show the sensitivity of the number of iterations required to the value of ω. The N-R and ADI [Fig. 2.3(b)] methods give wider range of relaxation factor than the I.L. method. The optimal relaxation factor of N-R, ADI, and I.L. for this experiment is found as 1.6, 1.5, and 1.45, respectively, based on the lowest number of iterations from each curve. Also, this example problem is studied for the effects of smoothness, orthogonality, and concentration, respectively. The weight function is specified as

$$W(x,z) = (x+z)[24-(x+z)] \quad . \tag{2.69}$$

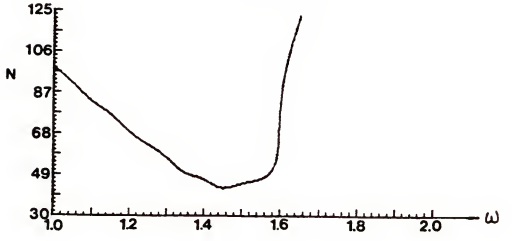
Figure 2.4(a)shows the result of minimizing the smoothness functional only ($I_\Omega = I_s$, $\lambda_o \equiv \frac{\lambda}{\lambda_o'} = 0$, $\lambda_w \equiv \frac{\lambda}{\lambda_w'} = 0$); it shows the grid points distributed smoothly. Figure 2.4(b) gives a result of minimizing the functionals of smoothness and orthogonality ($\lambda_o = 8$, $\lambda_w = 0$). Clearly, the orthogonality near the boundary is more strongly emphasized; however, the smoothness is not as good as in Fig. 2.4(a). Figure 2.4(c) shows a result of minimizing the functionals



(a) Newton-Raphson



(b) ADI



(c) Line Iteration

Fig. 2.3 Relaxation factor (ω) versus number of iterations (N)

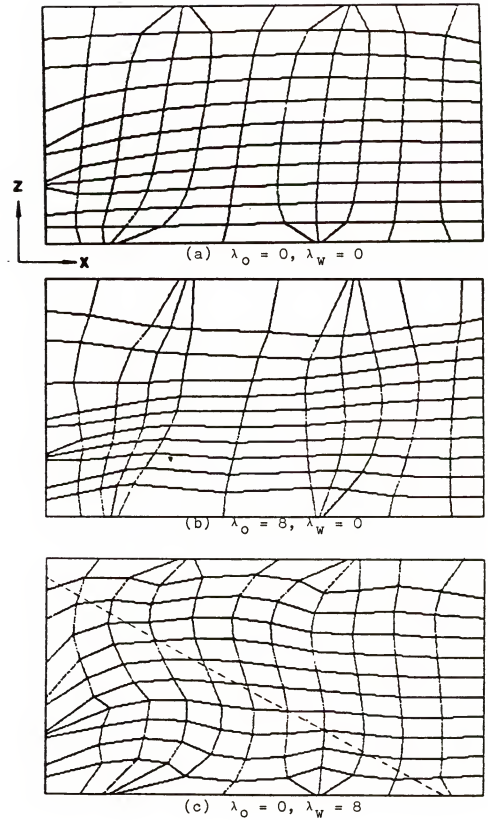


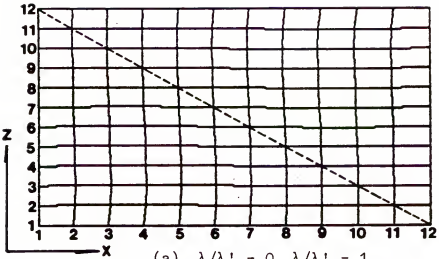
Fig. 2.4 Grid generations with unequal spacing on boundary

of smoothness and concentration ($\lambda_o = 0$, $\lambda_w = 8$). Since the maximum values of W(x,z) are on the line x+z = 12 [see the dashed line in Fig. 2.4(c)], it emphasizes the concentration more strongly than the region which is away from this line. Also, the grid in Fig. 2.4(c) is skewed in comparison with the grid in Fig. 2.4(a); however, if the adaptive boundary grid were applied to this boundary, the skewness would be reduced. Thus, the different control parameters result in different properties of the grid distributions. More detailed descriptions concerning the interaction of these parameters and changes can be found in references [6] and [19].

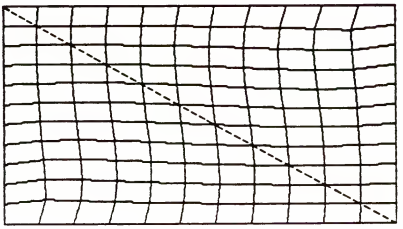
The second example problem, grid generation with uniform grid spacing, examines the influence of concentration on the grid network. The weight function is specified as

$$W(x,z) = (x+z)[26-(x+z)] \tag{2.70}$$

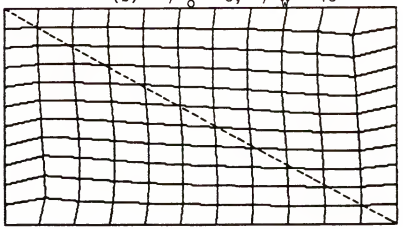
The maximum values of W(x,z) are on the line x+z = 13 and W(x,z) is decreased away from this line. Large values of W(x,z) cause higher concentration, smaller cells, and small values cause lower concentration. Figure 2.5(a), (b), and (c) show the results of minimizing the functionals of the smoothness with their concentrations λ_w = 1, 10, and 100, respectively. The grid generated in Fig. 2.5(b) is clearly different from Fig. 2.5(a), because larger values of the



(a) $\lambda/\lambda'_o = 0$, $\lambda/\lambda'_w = 1$



(b) $\lambda/\lambda'_o = 0$, $\lambda/\lambda'_w = 10$



(c) $\lambda/\lambda'_o = 0$, $\lambda/\lambda'_w = 100$

Fig. 2.5 Grid network with different λ_w and uniform grid on boundary

specified λ_w place more emphasis on concentration. However, the difference between Fig. 2.5(b) and 2.5(c) is not as pronounced. This indicates that the variation of the concentration functional converges to some constant value (i.e., WJ^{-2} = const.) when $\lambda_w >> 1$. In Fig. 2.5(c), the ratio of maximum value of W(x,z), $W_{max}(x,z)$ = 169, to minimum value, $W_{min}(x,z)$ = W(2,2) = W(11,11) = 88, is 169/88. The corresponding ratio of minimum to maximum values of cell area, $A_{min}/A_{max} = (88/169)^{1/2} \approx 0.72$, is close to 1 that is why the grid sizes near and far from the line x+z = 13 may not be clearly apparent in Fig. 2.5.

In order to analyze the efficiency and accuracy of the results, the average error, E, is defined as

$$E = [\Sigma(x-x_o)_{j,k}+\Sigma(z-z_o)_{j,k}]/2[(M-2)(N-2)] \qquad (2.71)$$

where j = 2, 3, . . ., M-1; k = 2, 3, . . ., N-1; the notation Σ represents the summation on both j and k; and $(x_o,z_o)_{j,k}$ are the convergent solutions found with the allowable error, $R_o = 10^{-7}$, while the approximate solutions $(x,z)_{j,k}$ correspond to some (considered convergent) larger allowable R_o. The first problem with $\lambda_o \equiv \lambda/\lambda'_o = 1$, $\lambda_w \equiv \lambda/\lambda'_w = 1$ is used for this experimental analysis. Figure 2.6 illustrates the efficiency analysis for those methods, which have the property that CPU time is linearly related to R_o; the slope of the line of the Jacobi Method (without update values and SOR procedure in N-R method) is highest

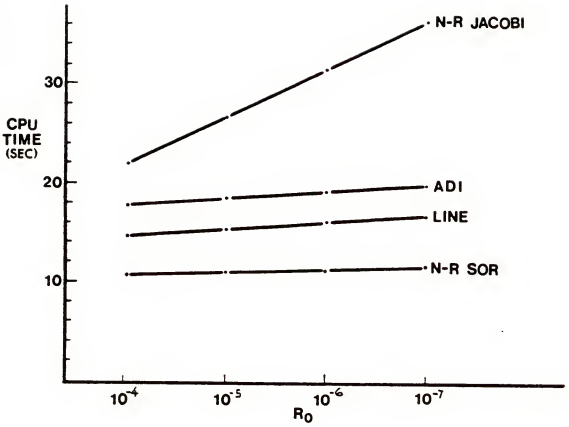


Fig. 2.6 Efficiency analysis

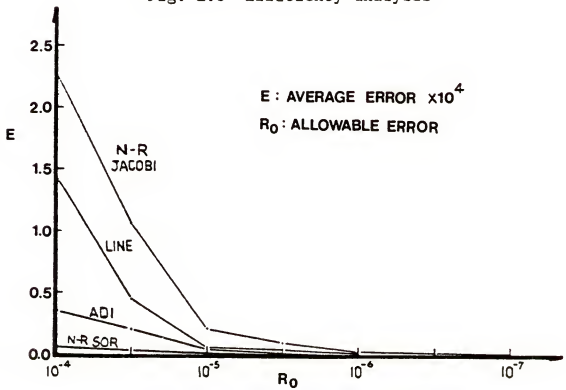


Fig. 2.7 Accuracy analysis

and N-R with SOR results in the lowest value of CPU time and slope in comparison with the others. Figure 2.7 illustrates the relations between average error E and R_o, it shows clearly that the larger differences in the values of E for the different methods occur at $R_o = 10^{-4}$ and that the differences are smaller up to $R_o = 10^{-5}$; N-R method with SOR results in the lowest E over all R_o up to 10^{-4}. However, usually, for most of the large number of grid generation, the allowable error R_o up to 10^{-4} is good enough, since the grid generation does not need to reach too highly accurate grid solution; also, a large CPU time would be expended to obtain a greater accuracy. In the view of efficiency and accuracy, we conclude that the N-R method with SOR is the most suitable of the methods examined to be used for the adaptive grid generation.

CHAPTER III
NAVIER-STOKES EQUATIONS

III.1 Review of Fundamental Principles

The purpose of this section is to review the set of governing equations resulting from the physical laws of conservation of mass, momentum, and energy for three-dimensional, compressible fluid flow.

The continuity equation, a consequence of the conservation of mass, is

$$\frac{\partial \rho}{\partial t} + \{\, \partial_x \; \partial_y \; \partial_z \,\} \{\, \rho u \; \rho v \; \rho w \,\}^T = 0 \tag{3.1}$$

where $\rho = \rho(x,y,z,t)$ is density, $u = u(x,y,z,t)$, $v = v(x,y,z,t)$, and $w = w(x,y,z,t)$ are the Cartesian velocity components. The notations $\partial_x = \partial/\partial_x$, . . ., etc., are partial derivatives.

The Navier-Stokes' equations, a consequence of the conservation of momentum, are

$$\rho d_t \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} = \begin{Bmatrix} \partial_x \\ \partial_y \\ \partial_z \end{Bmatrix}^T \begin{bmatrix} \sigma'_x - p & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma'_y - p & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma'_z - p \end{bmatrix}, \tag{3.2}$$

in which $d_t = {}^{d}/dt$ is a substantial derivative which consists of a local and convective contribution, p is the pressure, and the stress components for Newtonian fluid are

$$\begin{bmatrix} \sigma'_x & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma'_y & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma'_z \end{bmatrix} = \begin{bmatrix} \lambda(u_x+v_y+w_z)+2\mu u_x & \mu(u_y+v_x) & \mu(u_z+w_x) \\ \mu(v_x+u_y) & \lambda(u_x+v_y+w_z)+2\mu v_y & \mu(v_z+w_y) \\ \mu(w_x+u_z) & \mu(w_y+v_z) & \lambda(u_x+v_y+w_z)+2\mu w_z \end{bmatrix} \tag{3.3}$$

where $\lambda = -\frac{2}{3}\mu$ (no bulk viscosity) and μ are viscosity coefficients. The subscripts x, y, and z in the right hand side of Eqn. (3.3) denote the partial differentiations with respect to x, y, and z, respectively.

The energy equation, a consequence of the conservation of energy, can be written in the form

$$\frac{\partial e}{\partial t} + \begin{Bmatrix} \partial_x \\ \partial_y \\ \partial_z \end{Bmatrix}^T \begin{Bmatrix} eu \\ ev \\ ew \end{Bmatrix} = \begin{Bmatrix} \partial_x \\ \partial_y \\ \partial_z \end{Bmatrix}^T \begin{Bmatrix} \kappa T_x \\ \kappa T_y \\ \kappa T_z \end{Bmatrix} + \begin{Bmatrix} \partial_x \\ \partial_y \\ \partial_z \end{Bmatrix}^T \begin{bmatrix} \sigma'_x-p & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma'_y-p & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma'_z-p \end{bmatrix} \begin{Bmatrix} u \\ v \\ z \end{Bmatrix} \tag{3.4}$$

where e is the total energy per unit volume, given by

$$e = \rho[e_I + \frac{1}{2}(u^2+v^2+w^2)] \ , \tag{3.5}$$

and e_I is the internal energy per unit mass; κ is the coefficient of thermal conductivity, T is the temperature and T_x, T_y, and T_z are the partial derivatives of T with respect to x, y, and z, respectively.

For the perfect gas, the thermal and caloric equations of state can be defined as

$$p = \rho RT, \quad e_I = c_v T, \tag{3.6}$$

respectively, where R is the gas constant, and c_v is the specific heat at constant volume. Also, the following relationships exist

$$h = c_p T \ , \quad \gamma = \frac{c_p}{c_v} \ , \quad c_v = \frac{R}{\gamma-1} \ , \quad c_p = \frac{\gamma R}{\gamma-1} \tag{3.7}$$

where h is the enthalpy, c_p is the specific heat at constant pressure and γ is the ratio of specific heats. For air at standard conditions, $\gamma = 1.4$. Thus, the perfect gas equation of state can also be written as

$$p = (\gamma-1)[e - \frac{1}{2}\rho(u^2+v^2+w^2)] \tag{3.8}$$

The coefficients of viscosity and thermal conductivity have been related to the thermodynamic variables using kinetic theory. It is possible to use an interpolation

formula based on D.M. Sutherland's theory of viscosity [21]

$$\frac{\mu}{\mu_\infty} = \left(\frac{T}{T_\infty}\right)^{\frac{3}{2}} \left[\frac{1+S/T_\infty}{(T/T_\infty)+(S/T_\infty)}\right] \tag{3.9}$$

where μ_∞ denotes the viscosity at the reference temprature T_∞, and S is the Sutherland constant, which for air assumes the values S = 198.6°R or 110°K. The Prandtl number,

$$Pr = \frac{c_p \mu}{\kappa} , \tag{3.10}$$

is often used to determine the coefficient of thermal conductivity κ once μ is known. This is possible because the ratio (c_p/Pr) which appears in the expression

$$\kappa = \frac{c_p}{Pr} \mu \tag{3.11}$$

is approximately constant for most gases.

The derivatives of e_I in the caloric equation (3.6) with respect to x, y, and z at c_v = constant can be written as the vector form,

$$\{ T_x \; T_y \; T_z \} = \frac{1}{c_v} \{ \partial_x \; \partial_y \; \partial_z \} e_I$$

After substituting this equation and Eqn. (3.11) into Eqn. (3.4) and shifting the pressure terms to the left hand side,

the energy equation can be rewritten as the form

$$\partial_t e + \begin{Bmatrix} \partial_x \\ \partial_y \\ \partial_z \end{Bmatrix}^T \begin{Bmatrix} (e+p)u \\ (e+p)v \\ (e+p)w \end{Bmatrix} = \begin{Bmatrix} \partial_x \\ \partial_y \\ \partial_z \end{Bmatrix}^T \begin{Bmatrix} \beta_x \\ \beta_y \\ \beta_z \end{Bmatrix} \tag{3.12}$$

where

$$\begin{Bmatrix} \beta_x \\ \beta_y \\ \beta_z \end{Bmatrix} = \begin{Bmatrix} \gamma\mu Pr^{-1}\partial_x e_I + u\sigma'_x + v\tau_{yx} + w\tau_{zx} \\ \gamma\mu Pr^{-1}\partial_y e_I + u\tau_{xy} + v\sigma'_y + w\tau_{zy} \\ \gamma\mu Pr^{-1}\partial_z e_I + u\tau_{xz} + v\tau_{yz} + w\sigma'_z \end{Bmatrix} \tag{3.13}$$

III.2 Conservation-Law Form of Equations

A partial differential equation (PDE) can be put in conservation-law form, if the coefficients of the derivative terms are either constant or if variable, their derivatives disappear in the equation. In either case, the coefficients can be brought inside the derivative. Normally, physical conservation statements are represented by PDEs; this means that the divergence of a physical quantity can be identified in the equation [22]. Using an appropriate set of equations in conservation-law form represented by the PDEs of interest which all have their basis in physical laws such as the conservation of mass, momentum, and energy shows the nature of the physical conservation laws involved and allows overall integral properties to be maintained identically in

the finite-difference system. Such a system of equations is now in common use in calculations of very high gradient for a physical phenomenon, regardless of the finite-difference methods used, since the exact planar high gradient property such as shock speed will be produced by any stable method [23,24].

The continuity equation (3.1) and energy equation (3.12) are already in conservation-law forms. The conservation-law form of the momentum equation can be readily formed by adding the continuing equation (3.1) into Eqn. (3.2). The result is

$$\partial_t \begin{Bmatrix} \rho u \\ \rho v \\ \rho w \end{Bmatrix} + \begin{Bmatrix} \partial_x \\ \partial_y \\ \partial_z \end{Bmatrix}^T \begin{bmatrix} \rho u^2+p & \rho vu & \rho wu \\ \rho uv & \rho v^2+p & \rho wv \\ \rho uw & \rho vw & \rho w^2+p \end{bmatrix}$$

$$= \begin{Bmatrix} \partial_x \\ \partial_y \\ \partial_z \end{Bmatrix}^T \begin{bmatrix} \sigma'_x & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma'_y & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma'_z \end{bmatrix} \qquad (3.14)$$

Also, the compressible Navier-Stokes equations in Cartesian coordinates can be formed by assembling Eqn. (3.1), (3.14), and (3.12) into the compacted form

$$\partial_t \hat{q} + \partial_x \hat{f}_x + \partial_y \hat{f}_y + \partial_z \hat{f}_z = 0 \qquad (3.15)$$

where $\hat{q}$, $\hat{f}_x$, $\hat{f}_y$, and $\hat{f}_z$ are the five-component vectors

$$\hat{q} = \{\rho \ \rho u \ \rho v \ \rho w \ e\}^T$$

$$\hat{f}_x = \begin{Bmatrix} \rho u \\ \rho u^2 + p - \sigma'_x \\ \rho vu - \tau_{xy} \\ \rho wu - \tau_{xz} \\ (e+p)u - \beta_x \end{Bmatrix}, \quad \hat{f}_y = \begin{Bmatrix} \rho v \\ \rho uv - \tau_{yx} \\ \rho v^2 + p - \sigma'_y \\ \rho wv - \tau_{yz} \\ (e+p)v - \beta_y \end{Bmatrix}, \quad \hat{f}_z = \begin{Bmatrix} \rho w \\ \rho uw - \tau_{xz} \\ \rho vw - \tau_{zy} \\ \rho w^2 + p - \sigma'_z \\ (e+p)w - \beta_z \end{Bmatrix}$$

The governing equations can be made dimensionless. The viscosity μ_∞, the density ρ_∞, the temperature T_∞, the sound speed a_∞, and a characteristic length D are chosen as reference quantities, where the subscript ∞ denotes the free stream condition. Let

$$x^* = \frac{x}{D}, \quad y^* = \frac{y}{D}, \quad z^* = \frac{z}{D}, \quad t^* = \frac{t}{D/a_\infty}, \quad \mu^* = \frac{\mu}{\mu_\infty}, \quad u^* = \frac{u}{a_\infty},$$

$$v^* = \frac{v}{a_\infty}, \quad w^* = \frac{w}{a_\infty}, \quad \rho^* = \frac{\rho}{\rho_\infty}, \quad T^* = \frac{T}{T_\infty}, \quad e_I^* = \frac{e_I}{\rho_\infty a_\infty^2} \tag{3.16}$$

in which the superscript * denotes the nondimensional properties. Substituting Eqn. (3.16) into Eqn. (3.15) and finally dropping the superscript * lead to the following

dimensionless form of the equations

$$\partial_t q + \partial_x(E-E_v) + \partial_y(F-F_v) + \partial_z(G-G_v) = 0 \tag{3.17}$$

where q, E, F, and G are five-component vectors,

$$q = \begin{Bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{Bmatrix}, \quad E = \begin{Bmatrix} \rho u \\ p+\rho u^2 \\ \rho vu \\ \rho wu \\ (e+p)u \end{Bmatrix}, \quad F = \begin{Bmatrix} \rho v \\ \rho uv \\ p+\rho v^2 \\ \rho wv \\ (e+p)v \end{Bmatrix}, \quad G = \begin{Bmatrix} \rho w \\ \rho uw \\ \rho vw \\ p+\rho w^2 \\ (e+p)w \end{Bmatrix} \tag{3.18}$$

and E_v, F_v, and G_v are the viscous flux terms,

$$E_v = Re^{-1}\begin{Bmatrix} 0 \\ \sigma_x' \\ \tau_{xy} \\ \tau_{xz} \\ \beta_x \end{Bmatrix}, \quad F_v = Re^{-1}\begin{Bmatrix} 0 \\ \tau_{yx} \\ \sigma_y' \\ \tau_{yz} \\ \beta_y \end{Bmatrix}, \quad G_v = Re^{-1}\begin{Bmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_z' \\ \beta_z \end{Bmatrix} \tag{3.19}$$

The Reynolds number Re is defined as

$$Re = \rho_\infty a_\infty D/\mu_\infty \tag{3.20}$$

III.3 Transformed Governing Equations

As mentioned previously, it is possible to develop the well known "boundary-fitted curvilinear coordinate systems" by coordinate transformation so that unequal spacing in physical domain can be mapped onto equal spacing in the computational domain. The governing equations need to be transformed in the same manner. With this procedure, the solution of a partial differential system is calculated in a fixed parallelepiped field with a cubic grid in the computational domain. Also, the significant aspect of the transformation offered in Chapter II is that grid points can be adaptively clustered in regions that have rapid changes in the flow field gradients.

Let us now consider a general transformation of the three-dimensional form

$$\begin{aligned} \xi &= \xi(x,y,z,t) \\ \eta &= \eta(x,y,z,t) \\ \zeta &= \zeta(x,y,z,t) \\ \tau &= t \end{aligned} \tag{3.21}$$

where τ is the computational time. It can then be used to transform the governing equations from the physical domain (x,y,z,t) to the computation domain (ξ,η,ζ,τ). The three-dimensional version of Eqn. (2.26) is

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x_x & x_y & x_z \\ y_x & y_y & y_z \\ z_x & z_y & z_z \end{bmatrix} = \begin{bmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{bmatrix} \begin{bmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{bmatrix} \quad (3.22)$$

therefore, the metrics ξ_x, η_x, . . ., etc. are found by the cofactor method,

$$\begin{bmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{bmatrix} = \begin{bmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{bmatrix}^{-1}$$

$$= \frac{1}{J^{-1}} \begin{bmatrix} y_\eta z_\zeta - y_\zeta z_\eta & -(x_\eta z_\zeta - x_\zeta z_\eta) & x_\eta y_\zeta - x_\zeta y_\eta \\ -(y_\xi z_\zeta - y_\zeta z_\xi) & x_\xi z_\zeta - x_\zeta z_\xi & -(x_\xi y_\zeta - x_\zeta y_\xi) \\ y_\xi z_\eta - y_\eta z_\xi & -(x_\xi z_\eta - x_\eta z_\xi) & x_\xi y_\eta - x_\eta y_\xi \end{bmatrix} \quad (3.23)$$

where J^{-1}, the 3-D version of the Jacobian determinant defined by Eqn. (2.15), is given by

$$J^{-1} = \frac{\partial(x,y,z)}{\partial(\xi,\eta,\zeta)} = \begin{vmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{vmatrix} \quad (3.24)$$

$$= x_\xi(y_\eta z_\zeta - y_\zeta z_\eta) - x_\eta(y_\xi z_\zeta - y_\zeta z_\xi) + x_\zeta(y_\xi z_\eta - y_\eta z_\xi) \quad (3.25)$$

Thus J^{-1} represents the cells volume in 3-D case. The

metrics can be readily obtained after finding the inverse transformation,

$$\begin{aligned} x &= x(\xi,\eta,\zeta,\tau) \\ y &= y(\xi,\eta,\zeta,\tau) \\ z &= z(\xi,\eta,\zeta,\tau) \\ t &= \tau \end{aligned} \tag{3.26}$$

If the (ξ,η,ζ,τ) system is obtained by solving the governing equations of grid generation (i.e., Euler's equations, as described in Chapter II Adaptive Grid Generation), we may solve any set of PDE on this physical coordinate system by solving the transformed equations in the computational domain [25]. Also, the transformed computational coordinates of a given moving grid point in the physical space are fixed; thus,

$$\frac{d\xi}{d\tau} = \frac{d\eta}{d\tau} = \frac{d\zeta}{d\tau} \equiv 0 \tag{3.27}$$

gives

$$\begin{Bmatrix} \xi_t \\ \eta_t \\ \zeta_t \end{Bmatrix} = -\begin{Bmatrix} x_\tau \\ y_\tau \\ z_\tau \end{Bmatrix}^T \begin{bmatrix} \xi_x & \eta_x & \zeta_x \\ \xi_y & \eta_y & \zeta_y \\ \xi_z & \eta_z & \zeta_z \end{bmatrix} \tag{3.28}$$

where the subscripts t and τ denote partial differentiations with respect to physical time t and computational time τ, respectively.

We apply this generalized transformation to Eqn. (3.17) and obtain the following transformed equation

$$
\begin{aligned}
&q_\tau + \xi_t q_\xi + \eta_t q_\eta + \zeta_t q_\zeta \\
&+ \xi_x(E-E_v)_\xi + \eta_x(E-E_v)_\eta + \zeta_x(E-E_v)_\zeta \\
&+ \xi_x(F-F_v)_\xi + \eta_x(F-F_v)_\eta + \zeta_x(F-F_v)_\zeta \\
&+ \xi_x(G-G_v)_\xi + \eta_x(G-G_v)_\eta + \zeta_x(G-G_v)_\zeta = 0
\end{aligned}
\tag{3.29}
$$

This equation may be written with the metric coefficients inside the differentiation, thus producing a source term. The resulting form is called the "weak conservation law form"

$$
\begin{aligned}
&q_\tau + \partial_\xi[\xi_t q + \xi_x(E-E_v) + \xi_y(F-F_v) + \xi_z(G-G_v)] \\
&+ \partial_\eta[\eta_t q + \eta_x(E-E_v) + \eta_y(F-F_v) + \eta_z(G-G_v)] \\
&+ \partial_\zeta[\zeta_t q + \zeta_x(E-E_v) + \zeta_y(F-F_v) + \zeta_z(G-G_v)] + S' = 0
\end{aligned}
\tag{3.30}
$$

where

$$
\begin{aligned}
S' = &-[(\xi_t)_\xi + (\eta_t)_\eta + (\zeta_t)_\zeta]q - [(\xi_x)_\xi + (\eta_x)_\eta + (\zeta_x)_\zeta](E-E_v) \\
&-[(\xi_y)_\xi + (\eta_y)_\eta + (\zeta_y)_\zeta](F-F_v) \\
&-[(\xi_z)_\xi + (\eta_z)_\eta + (\zeta_z)_\zeta](G-G_v)
\end{aligned}
$$

It has been shown by Viviand [26] and Vinokur [27] that Eqn. (3.30) can be put back into "strong conservation-law

form." In order to do this, Eqn. (3.29) is first divided by the Jacobian and is then rearranged into conservation-law form by adding and subtracting like terms. The following equation results

$$\partial_\tau(J^{-1}q) + \partial_\xi\{J^{-1}[\xi_t q + \xi_x(E-E_v) + \xi_y(F-F_v) + \xi_z(G-G_v)]\}$$
$$+ \partial_\eta\{J^{-1}[\eta_t q + \eta_x(E-E_v) + \eta_y(F-F_v) + \eta_z(G-G_v)]\}$$
$$+ \partial_\zeta\{J^{-1}[\zeta_t q + \zeta_x(E-E_v) + \zeta_y(F-F_v) + \zeta_z(G-G_v)]\} + S = 0 \qquad (3.31)$$

where the source terms,

$$S = -[(J^{-1}\xi_t)_\xi + (J^{-1}\eta_t)_\eta + (J^{-1}\zeta_t)_\zeta]$$
$$- [(J^{-1}\xi_x)_\xi + (J^{-1}\eta_x)_\eta + (J^{-1}\zeta_x)_\zeta](E-E_v)$$
$$- [(J^{-1}\xi_y)_\xi + (J^{-1}\eta_y)_\eta + (J^{-1}\zeta_y)_\zeta](F-F_v)$$
$$- [(J^{-1}\xi_z)_\xi + (J^{-1}\eta_z)_\eta + (J^{-1}\zeta_z)_\zeta](G-G_v) \quad ,$$

are all equal to zero and can be dropped. This can be verified by substituting the metrics given by Eqn. (3.23) into this form. Therefore, Eqn. (3.31) leads to

$$\partial_\tau\hat{q} + \partial_\xi(\hat{E}-\hat{E}_v) + \partial_\eta(\hat{F}-\hat{F}_v) + \partial_\zeta(\hat{G}-\hat{G}_v) = 0 \qquad (3.32)$$

where $\hat{q}$, $\hat{E}$, $\hat{F}$, and $\hat{G}$ are

$$\hat{q} = J^{-1}\begin{Bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{Bmatrix}, \quad \hat{E} = J^{-1}\begin{Bmatrix} \rho U \\ \rho uU+\xi_x p \\ \rho vU+\xi_y p \\ \rho wU+\xi_z p \\ (e+p)U-\xi_t p \end{Bmatrix}, \quad \hat{F} = J^{-1}\begin{Bmatrix} \rho V \\ \rho uV+\eta_x p \\ \rho vV+\eta_y p \\ \rho wV+\eta_z p \\ (e+p)V-\eta_t p \end{Bmatrix},$$

$$\hat{G} = J^{-1}\begin{Bmatrix} \rho W \\ \rho uW+\zeta_x p \\ \rho vW+\zeta_y p \\ \rho wW+\zeta_z p \\ (e+p)W-\zeta_t p \end{Bmatrix}, \tag{3.33}$$

in which U, V, and W are contravariant velocities defined as

$$\begin{Bmatrix} U \\ V \\ W \end{Bmatrix} = \begin{Bmatrix} \xi_t \\ \eta_t \\ \zeta_t \end{Bmatrix} + \begin{bmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{bmatrix} \begin{Bmatrix} u \\ v \\ w \end{Bmatrix}$$

$$= \begin{bmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{bmatrix} \begin{Bmatrix} u-x_\tau \\ v-y_\tau \\ w-z_\tau \end{Bmatrix} \tag{3.34}$$

and the transformed viscous flux terms $\hat{E}_v$, $\hat{F}_v$, and $\hat{G}_v$ are given by

$$\hat{E}_v = (JRe)^{-1} \begin{Bmatrix} 0 \\ \xi_x \sigma'_x + \xi_y \tau_{xy} + \xi_z \tau_{zx} \\ \xi_x \tau_{yx} + \xi_y \sigma'_y + \xi_z \tau_{yz} \\ \xi_x \tau_{xz} + \xi_y \tau_{zy} + \xi_z \sigma'_z \\ \xi_x \beta_x + \xi_y \beta_y + \xi_z \beta_z \end{Bmatrix},$$

$$\hat{F}_v = (JRe)^{-1} \begin{Bmatrix} 0 \\ \eta_x \sigma'_x + \eta_y \tau_{xy} + \eta_z \tau_{zx} \\ \eta_x \tau_{yx} + \eta_y \sigma'_y + \eta_z \tau_{yz} \\ \eta_x \tau_{xz} + \eta_y \tau_{zy} + \eta_z \sigma'_z \\ \eta_x \beta_x + \eta_y \beta_y + \eta_z \beta_z \end{Bmatrix},$$

$$\hat{G}_v = (JRe)^{-1} \begin{Bmatrix} 0 \\ \zeta_x \sigma'_x + \zeta_y \tau_{xy} + \zeta_z \tau_{zx} \\ \zeta_x \tau_{yx} + \zeta_y \sigma'_y + \zeta_z \tau_{yz} \\ \zeta_x \tau_{xz} + \zeta_y \tau_{zy} + \zeta_z \sigma'_z \\ \zeta_x \beta_x + \zeta_y \beta_y + \zeta_z \beta_z \end{Bmatrix}. \qquad (3.35)$$

CHAPTER IV
THIN-LAYER NAVIER-STOKES EQUATIONS AND COMPUTER CODE

IV.1 Thin-Layer Navier-Stokes Equations

In the thin-layer approximation to the Navier-Stokes equations, the viscous terms containing derivatives in the directions parallel to the body surface are neglected in the unsteady Navier-Stokes equations, but all other terms in the three momentum equations are retained. One advantage of retaining those terms which are normally neglected in boundary-layer theory in the momentum equations is that separated and reverse flow regions can be readily computed. Also, flows which contain a high normal pressure gradient, such as the flow fields where the boundary-layer equations are not applicable, can be calculated [22]. Steger [10] was the first to consider a two-dimensional thin-layer model for simulation of flow about arbitrary geometrics with application to airfoils problem. Later, Baldwin and Lomax [28] evolved the thin-layer approximation from a detailed investigation of very high Reynolds number computation from the full Navier-Stokes equation. In these computations, a substantial fraction of the available computer storage and time is expended in resolving the normal gradients in the boundary layer since a

highly concentrated grid is required. As a result, the gradients parallel to the body surface are usually not resolved in an adequate manner even though the corresponding viscous terms are retained in the computations. Hence, for many Navier-Stokes computations, it makes sense to drop those terms provided that they are reasonably small. It should be emphasized that the thin-layer approximation is valid only for high Reynolds number flow.

For the application of the thin-layer approximation, it becomes necessary to map the physical body surface onto a transformed coordinate surface (i.e., ζ = const.). Thus, the transformed governing equation (3.32) can be simplified by dropping all viscous derivatives in the direction parallel to the body surface (ζ = const.), and we obtain the thin-layer Navier-Stokes equations,

$$\partial_\tau \hat{q} + \partial_\xi \hat{E} + \partial_\eta \hat{F} + \partial_\zeta \hat{G} = Re^{-1} \partial_\zeta \hat{S} \tag{4.1}$$

where $\hat{q}$, $\hat{E}$, $\hat{F}$, and $\hat{G}$ were defined by Eqn. (3.33) while the viscous terms $\hat{S}$ are readily obtained by retaining the viscous flux terms in the ζ-direction in $\hat{G}_v$ of the Eqn. (3.35).

$$\hat{S} = J^{-1} \begin{Bmatrix} 0 \\ \mu(\zeta_x^2+\zeta_y^2+\zeta_z^2)u_\zeta+(\mu/3)(\zeta_x u_\zeta+\zeta_y v_\zeta+\zeta_z w_\zeta)\zeta_x \\ \mu(\zeta_x^2+\zeta_y^2+\zeta_z^2)v_\zeta+(\mu/3)(\zeta_x u_\zeta+\zeta_y v_\zeta+\zeta_z w_\zeta)\zeta_y \\ \mu(\zeta_x^2+\zeta_y^2+\zeta_z^2)w_\zeta+(\mu/3)(\zeta_x u_\zeta+\zeta_y v_\zeta+\zeta_z w_\zeta)\zeta_z \\ \{(\zeta_x^2+\zeta_y^2+\zeta_z^2)[\frac{1}{2}\,\mu(u^2+v^2+w^2)_\zeta+\mu Pr^{-1}(\gamma-1)^{-1}(a^2)_\zeta \\ +(\mu/3)(\zeta_x u+\zeta_y v+\zeta_z w)(\zeta_x u_\zeta+\zeta_y v_\zeta+\zeta_z w_\zeta)\} \end{Bmatrix} \tag{4.2}$$

IV.2 Surface Boundary Conditions

For the boundary conditions along the surface $\zeta(x,y,z,t) = \text{const.}$, the velocities u, v, and w are obtained from Eqn. (3.34), such that

$$\begin{Bmatrix} u \\ v \\ w \end{Bmatrix} = J^{-1} \begin{bmatrix} (\eta_y\zeta_z-\eta_z\zeta_y) & -(\xi_y\zeta_z-\xi_z\zeta_y) & (\xi_y\eta_z-\xi_z\eta_y) \\ -(\eta_x\zeta_z-\eta_z\zeta_x) & (\xi_x\zeta_z-\xi_z\zeta_x) & -(\xi_x\eta_z-\xi_z\eta_x) \\ (\eta_x\zeta_y-\eta_y\zeta_x) & -(\xi_x\zeta_y-\xi_y\zeta_x) & (\xi_x\eta_y-\xi_y\eta_x) \end{bmatrix} \begin{Bmatrix} U-\xi_t \\ V-\eta_t \\ W-\zeta_t \end{Bmatrix} \tag{4.3}$$

In inviscid flow, W can be defined as $W \equiv 0$, while U and V are unknown values which can be found by extrapolation from interior points. The pressure along the body surface can be obtained from a normal momentum relation by combining those three transformed momentum equations.

$$P_n[\zeta_x^2+\zeta_y^2+\zeta_z^2]^{\frac{1}{2}}$$

$$= [\xi_x\zeta_x+\xi_y\zeta_y+\xi_z\zeta_z]P_\xi+[\eta_x\zeta_x+\eta_y\zeta_y+\eta_z\zeta_z]P_\eta+[\zeta_x^2+\zeta_y^2+\zeta_z^2]P_\zeta$$

$$= \rho[\partial_\tau\zeta_t+u\partial_\tau\zeta_x+v\partial_\tau\zeta_y+w\partial_\tau\zeta_z]-\rho U[\zeta_x u_\xi+\zeta_y v_\xi+\zeta_z w_\xi]$$

$$-\rho V[\zeta_x u_\eta+\zeta_y v_\eta+\zeta_z w_\eta] \qquad (4.4)$$

where P_n is the normal pressure gradient at the body surface and P_ξ, P_η, and P_ζ are the pressure gradients along the directions of ξ, η, ζ, respectively.

For viscous flows, the equations (4.3) and (4.4) are used with U = V = W = 0. All of the preceding boundary conditions are valid for steady or unsteady flow.

IV.3 Turbulence Model

The effects of turbulence can be simulated in terms of the eddy viscosity coefficient μ_t. The effective coefficient of viscosity μ_e is, therefore, defined by

$$\mu_e = \mu + \mu_t \qquad (4.5)$$

where μ is the dynamic laminar viscosity. Recently, the two-layer eddy viscosity closure model has been extensively used in solving practical turbulent flow problems. This model consists of inner and outer regions, consequently, the distribution of the two-layer effective eddy viscosity across the boundary layer can be written as

$$\mu_e = \begin{cases} (\mu_e)_i = \mu + (\mu_t)_i \\ (\mu_e)_o = \mu + (\mu_t)_o \end{cases} \tag{4.6}$$

in which subscripts i and o represent the inner and outer regions, respectively. Similarly, the effective heat flux term $\mu_e/(Pr)_e$ can be defined as

$$\frac{\mu_e}{(Pr)_e} = \frac{\mu}{Pr} + \frac{\mu_t}{(Pr)_t} \tag{4.7}$$

where $(Pr)_e$ and $(Pr)_t$ are viewed as effective and turbulent Prandtl numbers, respectively. Equations (4.6) and (4.7) are applied to the transformed thin-layer Navier-Stokes equation; thus, the μ and Pr in Eqn. (4.2) are replaced by μ_e and $(Pr)_e$, respectively.

The turbulence closure model programmed in the thin-layer Navier-Stokes code is an extension of the Cebci's algebraic eddy viscosity model for avoiding the finding of the boundary layer edge [28]. This model in terms of the two-layer algebraic eddy viscosity is given by

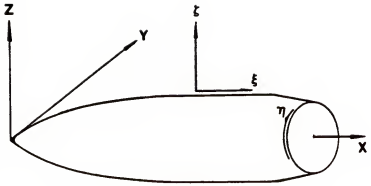
$$\mu_t = \begin{cases} (\mu_t)_i = \rho\{0.4z[1-\exp(-z^+/A^+)]\}^2 |\omega| \ , \ z \leq z_c \\ (\mu_t)_o = 0.0168\rho C_{cp} F_{wake} F_{kleb}(z) \ , \ z \leq z_c \end{cases} \tag{4.8}$$

where subscripts i and o represent the inner and outer regions, respectively. In Eqn. (3.16), z and ρ are the

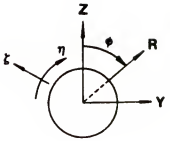
dimensionless quantities, the normal distance from the wall and the density, respectively; z_c is the smallest crossover value of z at which values from inner and outer formulas are equal. The constant A^+ denotes an empirical constant equal to 26. The law-of-the-wall coordinate z^+, vorticity $|\omega|$, Klebanoff intermittency factor $F_{kleb}(z)$, wake factor F_{wake}, and additional constant C_{cp} were defined in reference [28]. This model was applied to obtain the comparisons which were made with experiment for an incident shock on a flat plate, separated flow over a compression corner, and transonic flow over an airfoil. The results show that separation and reattachment points from numerical Navier-Stokes solutions agree with experiment within one boundary-layer thickness. Also, Pr and $(Pr)_t$ are defined by Pr = $(Pr)_t$ = 0.9 as reported in [28].

IV.4 Computer Code

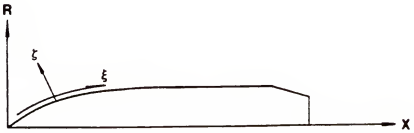
As described in the Introduction, Nietubicz, Pulliam, and Steger [14] developed a numerical procedure for an axisymmetric case form of the thin-layer Navier-Stokes equations. This axisymmetric thin-layer code with cylindrical coordinate system which has readily been applied to transonic projectile aerodynamics problems at the U.S. Army Ballistic Research Laboratory is more efficient than the thin-layer Navier-Stokes Code with Cartesian coordinate system. A sketch of a typical axisymmetric body is shown in Fig. 4.1(a). In order to determine the circumferential variations of typical flow and



(a) Projectile type body



(b) x = const. plane



(c) ϕ = const. plane

Fig. 4.1 Axisymmetric body and coordinate system

geometric parameters, we first establish correspondence between the Cartesian coordinates (x,y,z), the cylindrical coordinates (x,ϕ,R) in the physical space, and the transformed variables (ξ,η,ζ). For the cylindrical coordinates, it may be useful to have the reference line for ϕ rotate relative to the (x,y,z) system, so that $\phi = \phi(\tau)$, where τ is the time in (ξ,η,ζ) system. For example, the cylindrical coordinate grid system may be chosen to rotate relative to a spinning body. From the views shown in Fig. 4.1, the relationship between the coordinate systems are observed to be

$$\begin{aligned} \phi &= c\eta \\ x &= x(\xi,\zeta,\tau) \\ y &= R(\xi,\zeta,\tau)\sin\phi \\ z &= R(\xi,\zeta,\tau)\cos\phi \\ t &= \tau \end{aligned} \tag{4.9}$$

where c is an arbitrary constant which can be set equal to one, and ϕ is defined as $\phi = \phi(\tau)$. Note that x and R are functions of ξ, ζ, and time τ only. Thus, the metric terms are readily evaluated from Eqn. (4.9), such that

$$\begin{bmatrix} x_\xi & x_\eta & x_\zeta & x_\tau \\ y_\xi & y_\eta & y_\zeta & y_\tau \\ z_\xi & z_\eta & z_\zeta & z_\tau \end{bmatrix} = \begin{bmatrix} x_\xi & 0 & x_\zeta & x_\tau \\ R_\xi\sin\phi & \phi_\eta R\cos\phi & R_\zeta\sin\phi & R_\tau\sin\phi+\phi_\tau R\cos\phi \\ R_\xi\cos\phi & -\phi_\eta R\sin\phi & R_\zeta\cos\phi & R_\tau\cos\phi-\phi_\tau R\sin\phi \end{bmatrix} \tag{4.10}$$

Now substituting Eqn. (4.10) into Eqn. (3.23) and (3.28) yields

$$\begin{bmatrix} \xi_x & \xi_y & \xi_z & \xi_t \\ \eta_x & \eta_y & \eta_z & \eta_t \\ \zeta_x & \zeta_y & \zeta_z & \zeta_t \end{bmatrix}$$

$$= \frac{1}{J^{-1}} \begin{bmatrix} \phi_\eta RR_\zeta & \phi_\eta Rx_\zeta \sin\phi & \phi_\eta Rx_\zeta \cos\phi & \phi_\eta R(x_\zeta R_\tau - x_\tau R_\zeta) \\ 0 & \cos\phi/(JR\phi_\eta) & -\sin\phi/(JR\phi_\eta) & -\phi_\tau/(J\phi_\eta) \\ -\phi_\eta RR_\xi & \phi_\eta Rx_\xi \sin\phi & \phi_\eta Rx_\xi \cos\phi & \phi_\eta R(x_\tau R_\xi - R_\tau x_\xi) \end{bmatrix} \tag{4.11}$$

in which the Jacobian is given by

$$J^{-1} = R\phi_\eta (x_\xi R_\zeta - x_\zeta R_\xi) \tag{4.12}$$

After the term $\partial_\eta \hat{F}$ is evaluated in Eqn. (4.1), we can eliminate the cylindrical coordinates altogether by letting $\phi = 0^o$, hence R = z (see Fig. 4.1b), $\sin\phi = 0$ and $\cos\phi = 1$. The term ϕ_η is simply the scaling constant, $\phi_\eta = 1$, and the Cartesian velocities defined in Eqn. (4.3) become

$$\begin{Bmatrix} u \\ v \\ w \end{Bmatrix} = \begin{bmatrix} x_\xi & 0 & x_\zeta \\ R_\xi \sin\phi & R\phi_\eta \cos\phi & R_\zeta \sin\phi \\ R_\xi \cos\phi & -R\phi_\eta \sin\phi & R_\zeta \cos\phi \end{bmatrix} \begin{Bmatrix} U-\xi_t \\ V-\eta_t \\ W-\zeta_t \end{Bmatrix} \tag{4.13}$$

It should be emphasized, for an axisymmetric system, the state variables and the contravariant velocities, U, V, and

W are required to be invariant in the η-direction. The metrics terms ξ_t, η_t, and ζ_t are zero for steady non-spinning projectile flows and also are invariant in the η-direction for unsteady flow caused by body motion such as axial acceleration or axisymmetric spinning.

The resulting expressions for u, v, w, and the metrics, Eqn. (4.11), have only sinϕ and cosϕ variation in the η-direction. Consequently, Eqn. (4.11) and (4.13) can now be used in Eqn. (4.1) to reduce the η derivative of the flux vector $\hat{F}$ [Eqn. (3.33)], such that

$$\partial_\eta \hat{F} = \partial_\eta J^{-1} \begin{Bmatrix} \rho V \\ \rho uV + \eta_x p \\ \rho vV + \eta_y p \\ \rho wV + \eta_z p \\ (e+p)V - \eta_t p \end{Bmatrix} = J^{-1} \begin{Bmatrix} 0 \\ 0 \\ \rho V v_\eta + p(\eta_y)_\eta \\ \rho V w_\eta + p(\eta_z)_\eta \\ 0 \end{Bmatrix} \quad (4.14)$$

Transformation of the η derivative results in the following term:

$$\hat{H} \equiv \partial_\eta \hat{F} = J^{-1}\phi_\eta \begin{Bmatrix} 0 \\ 0 \\ \rho V[R_\xi(U-\xi_t)\cos\phi - R\phi_\eta(V-\eta_t)\sin\phi + R_\zeta(W-\zeta_t)\cos\phi] \\ \qquad - p\,\dfrac{\sin\phi}{R\phi_\eta} \\ -\rho V[R_\xi(U-\xi_t)\sin\phi - R\phi_\eta(V-\eta_t)\cos\phi + R_\zeta(W-\zeta_t)\sin\phi] \\ \qquad - p\,\dfrac{\cos\phi}{R\phi_\eta} \\ 0 \end{Bmatrix} \quad (4.15)$$

The resulting thin-layer n-invariant equations are written as

$$\partial_\tau \hat{q} + \partial_\xi \hat{E} + \partial_\zeta \hat{G} + \hat{H} = Re^{-1} \partial_\zeta \hat{S} \tag{4.16}$$

with the metrics and Cartesian velocities in $\hat{q}$, $\hat{E}$, $\hat{G}$, and $\hat{S}$ of Eqn. (4.1) evaluated at $\phi = 0^o$, and

$$\hat{H} = J^{-1} \phi_\eta \begin{Bmatrix} 0 \\ 0 \\ \rho V[R_\xi(U-\xi_t)+R_\zeta(W-\zeta_t)] \\ -\rho V R\phi_\eta(V-\eta_t)-p/(R\phi_\eta) \\ 0 \end{Bmatrix} \tag{4.17}$$

For surface boundary condition, the body surface pressure defined in Eqn. (4.4) is simplified as

$$\begin{aligned} P_n(\zeta_x^2+\zeta_z^2)^{\frac{1}{2}} &= (\xi_x\zeta_x+\xi_z\zeta_z)P_\xi+(\zeta_x^2+\zeta_z^2)P_\zeta \\ &= \rho[\partial_\tau\zeta_t+u\partial_\tau\zeta_x+v(J\phi_\eta Rx_\xi\phi_\tau)+w\partial_\tau(J\phi_\eta Rx_\xi)] \\ &\quad - \rho U(\zeta_x u_\xi+\zeta_z w_\xi)+\rho V[Jx_\xi R^2\phi_\eta^3(V-\eta_t)] \end{aligned} \tag{4.18}$$

where P_n is the normal pressure gradient at the body surface.

For an axisymmetric body spinning with angular velocity ω, we have the choice of using a (ξ,η,ζ) curvilinear coordinate grid in the physical space, so that body spins relative to the coordinate grid, or of having the grid system rotate with $-\omega$ relative to body. In the first case, the no slip condition would be

$$U = W = 0 \quad \text{and} \quad V = \omega \tag{4.19(a)}$$

In the second case, which is used in this study, the no slip boundary conditions is enforced by setting

$$U = V = W = 0 \tag{4.19(b)}$$

where U, V, and W are contravarient components of the relative velocity, relative to the body. Therefore, for the spinning body the usual no slip condition, Eqn. (4.19(b)), is imposed with the time metrics set at $\phi = 0^{o}$ (y=0) and consequently $\xi_y = \eta_z = \zeta_y = 0$ in Eqn. (4.11). Thus, Eqn. (3.28) is simplified as

$$\begin{aligned} \xi_t &= -y_\tau \xi_y - z_\tau \xi_z = -\omega(z\xi_y - y\xi_z) = 0 \\ \eta_t &= -y_\tau \eta_y - z_\tau \eta_z = -\omega(z\eta_y - y\eta_z) = -\omega/\phi_\eta \\ \zeta_t &= -y_\tau \zeta_y - z_\tau \zeta_z = -\omega(z\zeta_y - y\zeta_z) = 0 \end{aligned} \tag{4.20}$$

Once the contravariant velocities are specified at the body, the Cartesian velocities Eqn. (4.13) with $\phi = 0^{o}$ become

$$\begin{Bmatrix} u \\ v \\ w \end{Bmatrix} = J^{-1} \begin{bmatrix} x_\xi & 0 & x_\zeta \\ 0 & R\phi_\eta & 0 \\ R_\xi & 0 & R_\zeta \end{bmatrix} \begin{Bmatrix} U \\ V+\omega/\phi_\eta \\ 0 \end{Bmatrix} \tag{4.21}$$

The finite-difference scheme employed here is the implicit approximate factorization algorithm used in the delta form as analyzed by Warming and Beam [12]. An implicit

method was selected to avoid restrictive stability conditions which occur when small grid spacing is used. The solution of the two-dimensional system of equations (4.16) is implemented by an approximate factorization of the equations into two one-dimensional-like systems of equations. This procedure has been employed by Steger [10], Baldwin and Lomax [28], Pulliam and Steger [11], and Nietubicz [15]. Central-difference operators are used and the algorithm produces block tri-diagonal systems for each space coordinate. The stability and accuracy of the numerical algorithm for compressible Navier-Stokes equations are described in detail by Warming and Beam. The implicit approximation factorization algorithm was applied to the thin-layer approximation with transformed governing equations about arbitrary 2-D geometrics by Steger. Later, implicit finite-difference simulations of three-dimensional compressible flow were developed by Pulliam and Steger.

Using the Beam and Warming implicit algorithm applied to (4.16) results in the following finite difference equations

$$[I+h\delta_\xi \hat{A}^n-\varepsilon_I J^{-1}\Delta_\xi\nabla_\xi J][I+h\delta_\zeta \hat{C}^n-\varepsilon_I J^{-1}\Delta_\zeta\nabla_\zeta J-hRe^{-1}\delta_\zeta J^{-1}\hat{M}^n J]\Delta\hat{q}^n$$

$$= -\Delta t[\delta_\xi \hat{E}^n+\delta_\zeta \hat{G}^n-Re^{-1}\delta_\zeta \hat{S}^n]-\Delta t\hat{H}^n-\varepsilon_E J^{-1}[(\Delta_\xi\nabla_\xi)^2+(\Delta_\zeta\nabla_\zeta)^2]J\hat{q}^n \qquad (4.22)$$

where the δ's, Δ's, and ∇'s denote central, forward, and backward difference operators, respectively, e.g.,

$\Delta_\xi J = J(\xi+\Delta\xi,\zeta) - J(\xi,\zeta)$. Indices denoting spatial location are suppressed for convenience. The time step h is defined as $h = \Delta t$. The jacobian matrics $\hat{A} = \partial\hat{E}/\partial\hat{q}$, $\hat{C} = \partial\hat{G}/\partial\hat{q}$ along with the coefficient matrix $\hat{M}$ obtained from the local time linearization of $\hat{S}$ are described in detail by Steger [10]. The coefficients of the explicit and implicit "smoothing" terms are ϵ_E and ϵ_I, respectively; these are required to damp high frequency oscillations in the solution. A typical range recommended by Nietubicz et al. [14] is $\epsilon_E = (1 \text{ to } 5)\Delta t$ with $\epsilon_I = 2\epsilon_E$. We define $\Delta\hat{q}^n$ as

$$\Delta\hat{q}^n = \hat{q}^{n+1} - \hat{q}^n \ , \tag{4.23}$$

where $\hat{q}^n = \hat{q}(n\Delta t) = \hat{q}(t)$ is defined by letting the temporal variable t be discretized as $t = n\Delta t$.

The factored scheme (4.22) for the system of axisymmetric thin-layer Navier-Stokes Equations is, therefore, implemented as

$$[I+h\delta_\xi\hat{A}^n-\epsilon_I J^{-1}\Delta_\xi\nabla_\xi J]\Delta\hat{q}^* = \text{r.h.s.[Eqn.(4.22)]} \tag{4.24(a)}$$

$$[I+h\delta_\zeta\hat{C}^n-\epsilon_I J^{-1}\Delta_\zeta\nabla_\zeta J-hRe^{-1}\delta_\zeta J^{-1}\hat{M}^n J]\Delta\hat{q}^n = \Delta\hat{q}^* \tag{4.24(b)}$$

$$\hat{q}^{n+1} = \hat{q}^n + \Delta\hat{q}^n \tag{4.24(c)}$$

where $\Delta\hat{q}^*$ are intermediate temporal differences, and r.h.s. denotes right-hand-side.

Five subprograms, INITIA, EIGEN, STEP, MAP, and OUTPUT, are called in order in the MAIN program of this code. They are described in sequence as follows:

SUBROUTINE INITIA

This subroutine computes initial conditions and their relative input parameters. A few input parameters are read in this code; the detailed descriptions are given in Appendix A. The following set of constants are computed before generating grid and metrics,

RE=Re/1M, OMEGA=$2\pi\omega/60$, GAMMA=γ=1.4, SPIN=(PDOV)1M, GAMI=γ-1, GD=$\gamma(\gamma-1)$, HD=$\frac{1}{2}\Delta t$, JM=JMAX-1, LM=LMAX-1, ND2=(KMAX)(LMAX), DX1=DY1=DZ1=1.0, HDX=HD/DX1, HDY=HD/DY1, HDZ=HD/DZ1, CS=COS($\alpha\pi$/180), and SS=sin($\alpha\pi$/180),

where 1M is the Mach number, PDOV is a factor transferring Mach number to spin number, JMAX is the maximum number of grid points, along the ξ-direction, and α is the angle of attack which must be set equal to zero. The calculations of grid and metrics are implemented by calling the "SUBROUTINE GRID" and "SUBROUTINE METRIC," respectively. We then load the vector $\hat{q}_\infty$ that characterizes the free stream flow properties at α = 0. Thus,

$$\hat{q}_\infty = \{\rho, \rho u, \rho v, \rho w, e\}_\infty$$

$$= \{1.0, 1M\cos\alpha, 0, 1M\sin\alpha, \frac{1}{\gamma(\gamma-1)} + \frac{1}{2}\,1M^2\}_{\ell,j}\,,$$

$$\ell = 1, \ldots, \text{LMAX}, \; j=1, \ldots, \text{JMAX} \qquad (4.25)$$

where the total flow energy per unit volume, e_∞, can be derived from Eqn. (3.8),

$$p_\infty = (\gamma-1)[e_\infty - \frac{1}{2}\,1M_\infty^2] = \frac{\rho_\infty a_\infty^2}{\gamma} = \frac{1}{\gamma}$$

Thus,

$$e_\infty = \frac{1}{\gamma(\gamma-1)} + \frac{1}{2}\,1M_\infty^2$$

The turbulent viscosity coefficient and the source term are set to TURMU(L,J) = 0 and S(L,N,J) = 0, respectively, where N = 1~5 represents five fluid properties. There is an "IF" statement controlled by IREAD such that

IREAD { 0 ; no reading and passing over this statement for calculating the initial conditions
1 ; reading a new set of input for restart

The last process is to scale the variables by Jacobian

$$\hat{q}_{\ell,n,j} = \hat{q}_{\ell,n,j}/J_{\ell,j} = \hat{q}_{\ell,n,j}/\hat{q}_{\ell,6,j} \qquad (4.26)$$

where ℓ = 1, . . ., LMAX; n = 1, . . ., 5; and j = 1, . . ., JMAX.

SUBROUTINE EIGEN

This subroutine calculates the maximum Courant number, with CNBR = 0.0 as input, and the $\Delta\tau$ with CNBR $\neq$ 0.0 as

input. The purpose of this subroutine is to provide the information for the stability of the calculation.

SUBROUTINE STEP

An implicit integration scheme, Eqn. (4.24), is implemented by this subprogram. There are six subroutines, BC, RHS, SMOOTH, FILTRX, FILTRZ, and BTRI, called here. The step algorithm performs the computations described in order as follows:

(1) CALL SUBROUTINE BC

The subprogram BC calculates the fluid properties on the boundary. Since the outer boundary is in the free stream region, the values should remain in the free stream condition. For the inner boundary condition, the velocities are calculated by Eqn. (4.13), the body surface pressures obtained by the implementation of Eqn. (4.18), and the total flow energies per unit volume along body surface are found by Eqn. (3.8). The fluid properties on the axis boundary (AB line in Fig. 2.1) are obtained by making an extrapolation from interior points of the domain, and the same is done along the downstream boundary (DE line in Fig. 2.1). The values on the four segments of the boundary are, therefore, determined.

(2) CALL SUBROUTINE RHS

This subprogram constructs and calculates the vector form $\hat{S}_1$ of source term, one part of the right hand side of Eqn. (4.22),

$$\hat{S}_1 = -\Delta t[\delta_\xi \hat{E}^n + \delta_\zeta \hat{G}^n - Re^{-1}\delta_\zeta \hat{S}^n] - \Delta t\hat{H}^n$$

(3) CALL SUBROUTINE SMOOTH

In the other part of the right hand side of Eqn. (4.22), the explicit smoothing terms $\hat{S}_2$

$$\hat{S}_2 = -\varepsilon_E J^{-1}[(\Delta_\xi \nabla_\xi)^2 + (\Delta_\zeta \nabla_\zeta)^2]J\hat{q}^n$$

are executed.

(4) Once we have obtained the values of the source terms in Eqn. (4.22), the residual at the end of 10 time-step is computed by taking the root mean square of these summation values, such that

$$S = \sum_{j=k=\ell=1}^{JMAX,5,LMAX} [\hat{S}_1 + \hat{S}_2]^2_{j,k,\ell} \qquad \text{and}$$

$$\text{Residual} = \frac{1}{\Delta t}\sqrt{\frac{S}{(JMAX-2)(LMAX-2)}}$$

Thus, the residual is the mean residual over the whole field. This value is printed out for the inspection that gives some information on the accuracy of the results for the purpose of use.

(5) CALL SUBROUTINE FILTRX

This subprogram constructs and calculates the coefficients of the unknown vector $\Delta\hat{q}^*$ in Eqn. (4.24(a)); the

set of these coefficients is formed by a block tri-diagonal matrix with ξ-factor,

$$[\xi\text{-matrix}]^n = [I+h\delta_\xi \hat{A}^n - \varepsilon_I J^{-1}\nabla_\xi \Delta_\xi J]_\ell \ , \ \ell = 2, 3, \ldots, LM$$

(6) CALL SUBROUTINE BTRI

Thus, $\Delta\hat{q}^*$ is found calling this subprogram which is the block tri-diagonal solver.

(7) CALL SUBROUTINE FILTRZ

Since $\Delta\hat{q}^*$ becomes known vector, we then substitute $\Delta\hat{q}^*$ into Eqn. (4.24(b)). Similarly, this subprogram is called to evaluate a block tri-diagonal ζ-matrix, the coefficients of the unknown vector $\Delta\hat{q}$ in Eqn. (4.24(b)), formed as

$$[\zeta\text{-matrix}]^n = [I+h\delta_\zeta \hat{C}^n - \varepsilon_I J^{-1}\Delta_\xi \nabla_\xi J - hRe^{-1}\delta_\xi J^{-1}\hat{M}^n J] \ ,$$

$j = 2, \ldots, JM$

(8) CALL SUBROUTINE BTRI

The unknown vector $\Delta\hat{q}^n$ is found by calling BTRI once more. Thus, the $\hat{q}^{n+1}$ in Eqn. (4.24(c)) can be readily found.

SUBROUTINE MAP

This subroutine sketches the digital map of the calculated pressure distribution for assistance to inspect the behavior of pressure calculation roughly.

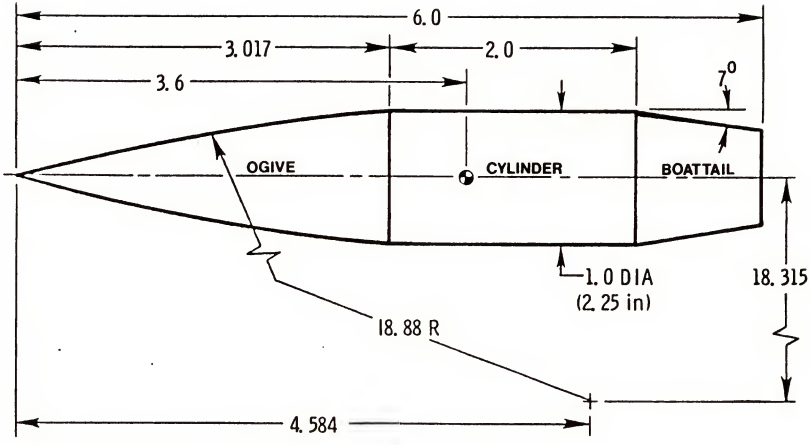
SUBROUTINE OUTPUT

The subroutine OUTPUT is designed to print out the distribution of fluid properties in the domain or on the boundary.

CHAPTER V
ADAPTIVE GRID GENERATION CODE

The geometry of a secant-ogive-cylinder-boattail (SOCBT) projectile is shown in Fig. 5.1; the model has a 3-caliber secant-ogive, a 2-caliber cylinder, and a 1-caliber 7^{o} boattail. Since the projectile is axisymmetric, we just need one-half the entire domain as shown in Fig. 2.1. The outer boundary is selected to keep free stream everywhere on this boundary. Behind the boattail there exists the wake flow which is not easy to solve. It is modeled as an attached "sting" instead of the wake for simplicity. The axis boundary (ξ = 1, AB line in Fig. 2.1) coincides with the model axis and the downstream boundary (ξ = M,DE line in Fig. 2.1) is perpendicular to the sting.

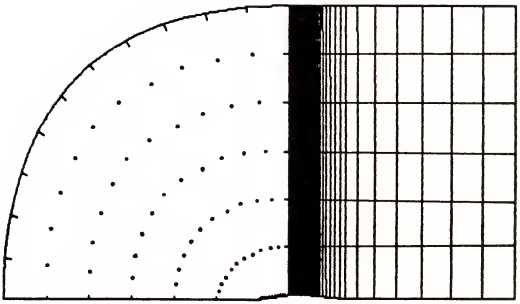
To illustrate the use of the code GRIDGEN mentioned in Chapter I, it is applied to the projectile flow problem for numerical experiment in this study. For example, the flow field has the computational domain extended to 4-times body length in front, 5 body lengths between the projectile axis and the straight line of outer boundary, and 3 body lengths behind the actual projectile. The sting is modeled (see Fig. 2.1) as a natural extension of the boattail for a distance of



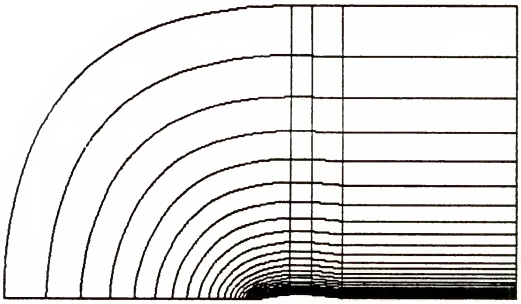
ALL DIMENSIONS IN CALIBERS

Fig. 5.1 Projectile configuration

1.767 calibers and then turned parallel to the model axis. Thus, the boundaries of the flow field are formed. There are 70x35 grid points used in this domain, 70 points along the ξ-direction (23 points on the ogive, 22 points on the cylinder, 17 points on the boattail, and 8 points on the sting), and 35 points along the ζ-direction. The grid points of the inner and outer boundaries are defined initially by using the geometrical analysis techniques for the projectile flow computation developed by Steger, Nietubicz, and Heavey [16]. Then, the grid points on the boundary are regenerated adaptively with a cubic function's clustering routine which permits the user to choose regions of clustering according to the user's experience and intuition. The grids in the first subdomain, the ogive part, are generated by using an elliptic solver; the initial locations of grid points in this subdomain are specified by equal spacing interpolation on the lines which are the connections between the inner boundary point and the corresponding outer boundary points [see the dot points in the first subdomain of Fig. 5.2(a)]. The other parts, cylinder, boattail, and sting, are the second subdomain which can be readily generated by the straight ray technique with the clustered grid points on the boundary as shown in the second subdomain in Fig. 5.2(a). The grid network is then generated; also, this grid network can be modified to emphasize high grid resolution in the viscous layer by clustering grid points with an exponential function along grid lines normal to the streamwise direction as shown

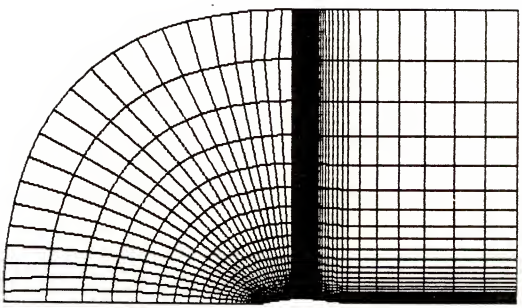


(a) Grid points clustering on the inner and outer boundaries by GRIDGEN



(b) Grid points clustering close to the inner boundary by GRIDGEN

Fig. 5.2 Grid networks generated by GRIDGEN



(c) Grid network generated by GRIDGEN

Fig. 5.2-continued.

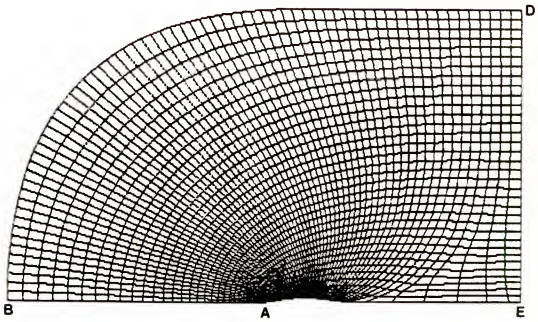


Fig. 5.3 Initial grid network generated by adaptive GRIDGEN

in Fig. 5.2(b). The resultant grid network generated by GRIDGEN is shown in Fig. 5.2(c).

As mentioned in the Introduction, the current version of GRIDGEN can generate a planar grid network by solving either an elliptic boundary value problem or a hyperbolic initial value problem. Also, Hsu [29] has provided an additional option to GRIDGEN for generating a grid network with the resolution of boundary grid adaptive to a specified control function such as a computed pressure gradient distribution instead of boundary grid adaptive to user's intuition. Moreover, he modified the hyperbolic grid routines of GRIDGEN for better smoothness and orthogonality of grid networks.

Using the same size of domain and the same number of corresponding grid points on the boundaries as used in GRIDGEN, the initial grid network adaptive to the control weight function is generated by using N-R method with SOR(ω = 1.6) to solve Eqn. (2.57) within the entire domain. Since no information is available initially for measuring the concentration of the grid, we set $\lambda_o = \lambda/\lambda'_o = 1$, $\lambda_w = \lambda/\lambda_w = 0$. The grid points on the inner and outer boundary are specified as either the uniform or the arbitrary grid spacing. Since experience shows that the grids on the axis (AB line in Fig. 5.3) and downstream (DE line) boundaries adaptive to the control function obtained from the solution of the thin-layer code do not give a good convergent solution, these grid points are determined by an

extrapolation of the interior grid curves to be orthogonal to the boundary. This extrapolating procedure is similar to that used to obtain the fluid properties on the boundary with the thin-layer code in Chapter IV. The iteration continues until either the solutions converge everywhere to within the allowable error or it is quitted at a specified maximum iteration number. The resulting initial grid network, for example, is shown in Fig. 5.3.

Once the approximate solution for the fluid properties is obtained from the calculation of 50~100 iterations by the thin-layer code based on the initial grid network; the grid generations adaptive to some property of the given solution, which is the pressure gradient used in this code as the weight function, are implemented with the scaled λ_w and λ_o to solve Eqn. (2.64) and (2.57) on the boundary and within the domain, respectively.

For successful adaptive grid network generation, a weight function is needed that is a measure of the steepness of the gradient of the important physical phenomenon. For example, a steep pressure gradient may occur at the corresponding location where a shock wave front exists in the flow field of the high speed aerodynamics problem. Thus, a norm of the pressure gradient, equal to the sum of the absolute values of the Cartesian components of the pressure gradient is used in the solutions given here, such that

$$W(x,z) = |P_x| + |P_z| \quad \text{in } \Omega \tag{5.1}$$

$$W(s) = |P_s| \quad \text{on } \Gamma \tag{5.2}$$

where $P_x = P_\xi \xi_x + P_\zeta \zeta_x$, $P_z = P_\xi \xi_z + P_\zeta \zeta_z$ are found by applying Eqn. (2.27) to the computational form

$$\begin{aligned} P_x &= J(z_\zeta P_\xi - z_\xi P_\zeta) \\ P_z &= J(x_\xi P_\zeta - x_\zeta P_\xi) \end{aligned} \tag{5.3}$$

The P_s in Eqn. (5.2) is the absolute value of pressure gradient along the boundary, such that $P_s = P_\xi / S_\xi$.

Since we discretize W(x,z) in central differences, it must be noted that the pressure gradient at peak points (i.e., maximum and minimum value of pressure) causes a large truncation error. Thus, the adaptive grid may not generate an optimum grid point in this region; the pressure gradient at certain peak points may be treated specially by taking averages in the vicinity of this point, such that

$$\begin{aligned} (P_\xi)_{j,k} &= \tfrac{1}{2}[|P_\xi|_{j+1,k} + |P_\xi|_{j-1,k}] \ ; \text{ and/or} \\ (P_\zeta)_{j,k} &= \tfrac{1}{2}[|P_\zeta|_{j,k+1} + |P_\zeta|_{j,k-1}] \end{aligned} \tag{5.4}$$

Also, because the finite difference equation roughens the data causing spurious changes for W(x,z), Brackbill and Saltzman suggested the smoothing equations,

$$W^{\ell+1}_{j,k} = W^{\ell}_{j,k} + \nu[(W^{\ell}_{j+1,k} + W^{\ell}_{j-1,k} + W^{\ell}_{j,k+1} + W^{\ell}_{j,k-1})/4 - W^{\ell}_{j,k}] \text{ in } \Omega \tag{5.5}$$

$$W^{\ell+1}_{j,k} = W^{\ell}_{j,k} + \nu[(W^{\ell}_{j+1,k} + W^{\ell}_{j-1,k})/2 - W^{\ell}_{j,k}] \text{ on } \Gamma \tag{5.6}$$

to modify the weight functions. These equations are executed with $\nu = 0.4$ for two or three iterations, which produces a good result.

To solve the Eqn. (2.48), the boundary grid system must be transformed from 2-D to 1-D. The line space, $S = S(x,z)$, is defined as

$$S^i_j = \sum_{j=1}^{M} [(x_{j+1,1} - x_{j,1})^2 + (z_{j+1,1} - z_{j,1})^2]^{\frac{1}{2}}$$

along inner Γ (5.7)

$$S_j = \sum_{j=1}^{M} [(x_{j+1,N} - x_{j,N})^2 + (z_{j+1,N} - z_{j,N})^2]^{\frac{1}{2}}$$

along outer Γ (5.8)

For inner Γ, there are four segments (ogive, cylinder, boattail, and sting) combined together, the superscript i is the number of the segment, for example, S^2_j is the line space on the cylinder. This treatment is required to fix the two end points of each segment in order to keep the projectile shape unvarying when implementing the iterations for boundary

adaptive grid generation. In other words, the adaptive grid for the inner boundary is executed separately, segment by segment. The S_j on outer Γ is implemented without any separation. Once we have solved for S_j, we go back to find the corresponding $(x_{j,1},z_{j,1})$ on the inner boundary and $(x_{j,N},z_{j,N})$ on the outer boundary by employing the cubic spline interpolation function.

The adaptive grid generation code with control weight function is primarily based on the code GRIDGEN. It is modified by the insertion of a few subroutines and provides a new option. The details of the new subroutines will be found in Appendix B. Also, it is convenient by coupling the adaptive GRIDGEN with control function and thin-layer code together to obtain a direct computation for projectile flow problem. One can then change the input, the new grid, and the time level automatically with a few control numbers.

CHAPTER VI
NUMERICAL EXPERIMENTS

As mentioned in Chapter V, the thin-layer code and adaptive grid generation code are coupled together for calculating the solution of the projectile flow problem. The thin-layer code provides the prior approximate solution properties for the control weight function of the adaptive grid generation code. This grid generation code then provides the adaptive grid network for the thin-layer code calculation of fluid properties. This cycle is repeated until a satisfactory solution is found.

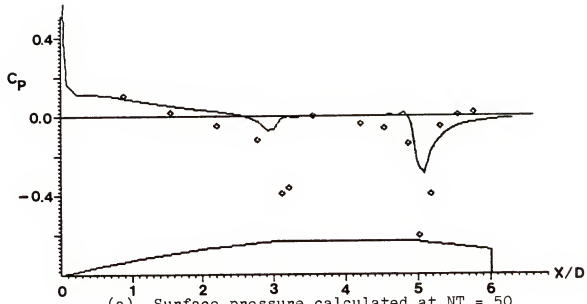
For assessing the effectiveness of the adaptive grid generation technique developed, we have considered the axisymmetric inviscid transonic projectile aerodynamics problems. The smoothing parameters are chosen as $\varepsilon_I = 2\varepsilon_E = 8\Delta t$ for this study. The weight function employed for grid resolution can be related to the solution of fluid properties such as the pressure, velocities, density and temperature. However, the pressure gradient seems to be, in some sense, a rather strong gradient of physical phenomena describing the location of shocks in the inviscid fluid flow.

Brackbill and Saltzman chose the high power of the physical gradient as the weight function, such as $(\frac{\nabla p}{p})^2$ in

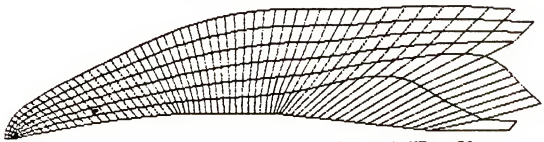
reference [8] for solving inviscid supersonic problems and $(\frac{\nabla\phi}{\phi})^4$ in reference [7] for a steady convection diffusion problem. For the choice of a proper weight function in the projectile flow field, we feel that the first order power of ∇p will be sufficient enough for emphasizing the concentration of grid. Therefore, the weight function used in this study is defined as

$$W(x,z) = |P_x| + |P_z| \ .$$

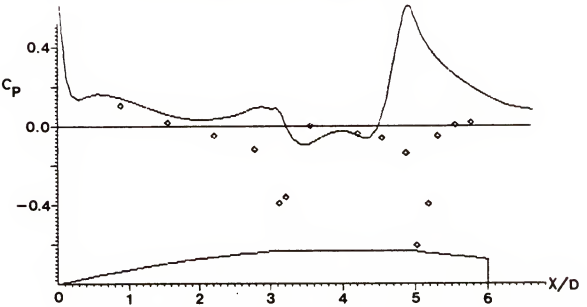
For conducting the numerical investigations on the adaptive grid generation technique, an experimental case for the calculation with the weight function, $W(x,z) = |P_x| + |P_z|$, and 1M = 0.91 was executed by the coupled computer codes. The constant penalty parameters $\lambda_w \equiv \lambda/\lambda'_w$ and $\lambda_o \equiv \lambda/\lambda'_o$ defined in Eqn. (2.20) were used in this case, in which λ'_w and λ'_o are constants similar to those of Saltzman which have been described in Chapter II, and $\lambda = 4.0$. The initial grid network used is the one shown in Fig. 5.3 (70x35 points). The computed surface pressure coefficient C_p at NT = 50 (NT stands for the number of time steps) is shown in Fig. 6.1(a); also, the measured data (diamonds) for steady flow obtained from reference [30] are plotted for comparison with the solution. A new grid network generated adaptively to the approximate solution at NT = 50 is shown in Fig.



(a) Surface pressure calculated at NT = 50



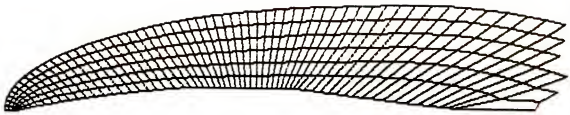
(b) Grid network near body surface at NT = 50



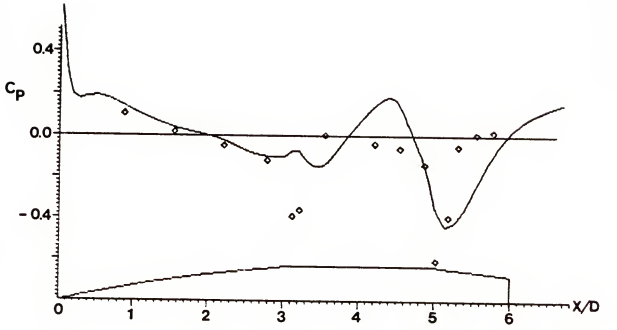
(c) Surface pressure calculated at NT = 200

Fig. 6.1 Surface pressure and adaptive grid networks

6.1(b). Subsequently, a new updated adaptive grid network is generated at every 150 time-step intervals. The computational process proceeds to NT = 350; however, in the next 150 time steps, the thin-layer Navier-Stokes code fails to converge. Figure 6.1(a) shows the surface pressure distribution for NT = 50, which tends to converge to the measured data. The grid network generated adaptively to the approximate solution at NT = 50 gives a large grid spacing near the region of the boattail as shown in Fig. 6.1(b). The next 150 time-step calculation gives the C_p-distribution at NT = 200 in Fig. 6.1(c). Apparently, the pressure near the large grid spacing region in Fig. 6.1(b) does not agree with the measured data. In fact, it has a positive peak where the measured data have a negative peak. These poor results probably come from the large grid spacing described in Fig. 6.1(b). Similarly, the grid spacing generated adaptively to the solution at NT = 200 is also large near the boattail region in Fig. 6.2(a). The surface pressure calculated at NT = 350 disagrees greatly with the measured data near that region in Fig. 6.2(b). Consequently, the grid spacing of the grid network generated adaptively to the solution at NT = 350 is still large in the same region as shown in Fig. 6.2(c). At last, these new grid networks generated adaptively to an unreasonable solution cause the computational instability. This large grid spacing may be caused by the smoothing effect, as occurred in conventional elliptic grids, and the small value of the control weight function in that region.



(a) Grid network generated at NT = 200



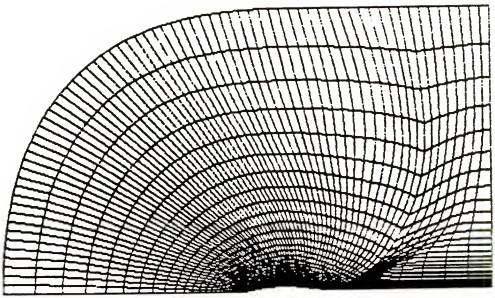
(b) Surface pressure at NT = 350

(c) Grid network generated at NT = 350

Fig. 6.2 Surface pressure and adaptive grid networks

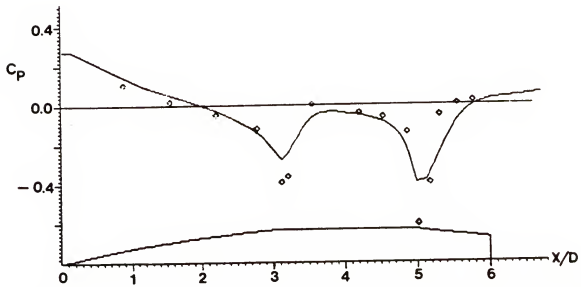
To overcome this difficulty, an exponential clustering similar to that of GRIDGEN is applied to all grid lines normal to the streamwise direction. The first grid spacing at the boundary is specified as $\Delta S = 0.01$. The resulting adaptive grid network with clustering is shown in Fig. 6.3(a) and the effect of the clustering can be seen by comparing Fig. 6.3(b) and Fig. 6.2(c). The application of this clustered grid network and the use of the poor results computed at NT = 350 to restart the computational process results in stable computation. The surface pressure coefficient computed after 150 time steps is clearly converging to the measured data as shown in Fig. 6.3(c). To confirm the importance of a clustering strategy, a new grid is generated at NT = 500 for the continuation of the integration process. The results obtained at NT = 650 and shown in Fig. 6.4(b) indicate that the implementation of the clustering strategy in the adaptive grid generation can provide a good grid network for the projectile aerodynamics computations.

The adaptive grid generation with clustering was next tested on the projectile aerodynamic problem at 1M = 0.96 to see whether a better grid network was obtained, which would give better results. The adaptive grid system is changed after each 200 time-step interval. The constant penalty parameters and weight function used are the same as in the previous experimental case. The C_p-distribution computed at



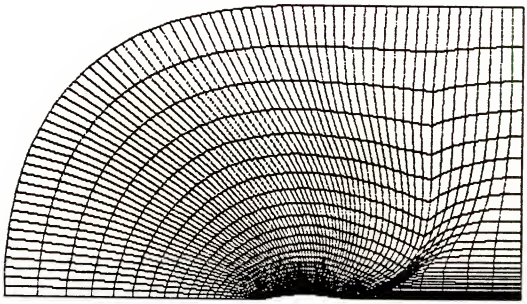
(a) Grid network with clustering (ΔS = 0.01) at NT = 350

(b) Expanded view of grids near the body surface

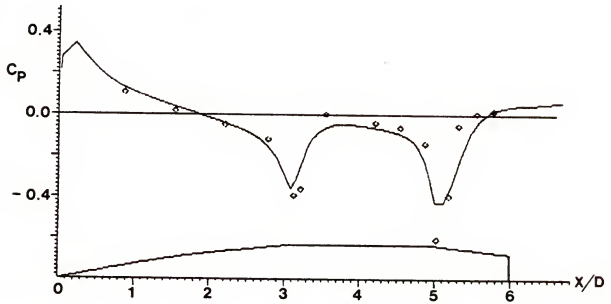


(c) Surface pressure calculated on the above grids at NT = 500

Fig. 6.3 Adaptive grid networks and surface pressure distribution



(a) Grid network generated at NT = 500 with clustering

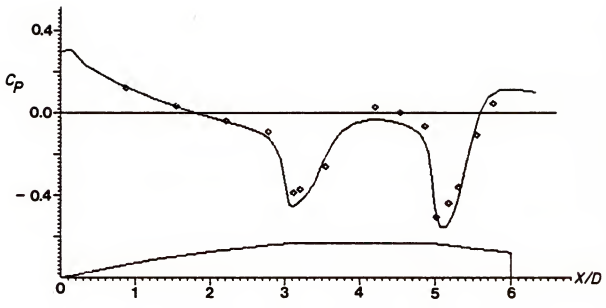


(b) Surface pressure at NT = 650

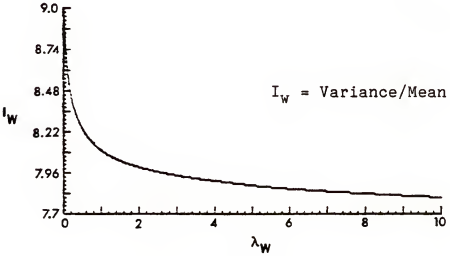
Fig. 6.4 Adaptive grid network and surface pressure

NT = 1650 is plotted in Fig. 6.5(a). It is clear that some values of the surface pressure C_p still do not approach closely to their corresponding experimental data. Apparently, we still cannot obtain a better adaptive grid network. In order to improve the adaptive grid network further, Brackbill and Saltzman paid much attention to modifying their weight function and to getting another type of weight function [31]. Also, high order power of physical gradient was chosen as weight function in their reports, as described previously. All of those treatments for the weight function were in order to emphasize the concentration of grids more strongly.

The most fruitful direction for obtaining a good grid system is probably in the determination of the penalty parameters. Unfortunately, the value of $\lambda'_w = \bar{W}(L/L')^4$ defined in Eqn. (2.24) within the domain is difficult to match to the values of $\lambda'_w = \bar{W}(L/L')^3$ given in Eqn. (2.45) on the boundaries when we apply these normalized constant penalty parameters to the 1-D variational principle for the adaptive boundary grid. Since the four side segments of the boundary have different lengths of L corresponding to the number of grid points L', the values of (L/L') for these segments are not equal. Consequently, to gain the consistency between boundary and domain, the value of (L/L') in the domain cannot be constant in the domain. Otherwise, if a reasonable value of (L/L'), such as the square root of the total grid area over the total grid points, is chosen



(a) Body surface pressure at NT = 1650 with constant penalty numbers



(b) The variance over the mean as a function of λ_W

Fig. 6.5 Surface pressure and the variance versus λ_W

for the domain the investigation of an optimal value of the global parameter λ in $\lambda_w = \lambda/\lambda'_w$ becomes tedious, and it may not give a good result. Therefore, the constant penalty parameters may not be suitable for generating a good adaptive grid network in both boundary and domain. The consistency of the grid points between boundary and domain thus becomes extremely important to obtain the accurate results.

In order to demonstrate substantially the properties of WJ^{-2} = const. in the projectile flow field, an experiment was performed to investigate the behavior of the variation of the concentration functional versus λ_w. In this experiment, λ_o is set to 1, and λ_w varies. The pressure gradient calculated at NT = 1650 is chosen as the weight function. We then use statistical analysis to find the dependence on λ_w of some measures of the distribution of WJ^{-2}. The mean and variance of the variation of WJ^{-2} are defined as

$$\text{mean} = [\sum_{j=k=1}^{M,N} (WJ^{-2})_{j,k}]/MN \tag{6.1}$$

$$\text{variance} = [\sum_{j=k=1}^{M,N} (WJ^{-2}-\text{mean})^2_{j,k}]/MN \tag{6.2}$$

Table 6.1 shows the variance/mean of the values WJ^{-2} as λ_w varies. These data are also plotted in Fig. 6.5(b) which clearly indicate that the variance decreases and approaches to the solution WJ^{-2} = const. as λ_w increases. These

Table 6.1
Numerical Experiment Data for Variance/Mean Versus λ_w

λ_w	Variance/Mean
0.00001	0.9034E+01
0.00005	0.9034E+01
0.00010	0.9034E+01
0.00050	0.9032E+01
0.00100	0.9030E+01
0.00500	0.9013E+01
0.01000	0.8993E+01
0.05000	0.8845E+01
0.10000	0.8704E+01
0.50000	0.8279E+01
1.00000	0.8139E+01
2.00000	0.8032E+01
4.00000	0.7943E+01
8.00000	0.7863E+01
10.00000	0.7839E+01

results may be helpful for finding adequate penalty parameters later on.

As discussed previously, λ_o' and λ_w' may be considered as the variables within the domain and on the boundaries in order to emphasize the orthogonality and concentration locally and obtain a good consistency of grids between boundary and domain. Thus, the calculations of L, L' and $\bar{W}$ may be redefined in each cell, such that

$$L = \Delta S(\xi,\zeta) = \begin{cases} |D\xi|_{j,k} = [(X_\xi)^2+(Z_\xi)^2]^{\frac{1}{2}}_{j,k} & \text{, along } \xi\text{-direction} \quad (6.3) \\ |D\zeta|_{j,k} = [(X_\zeta)^2+(Z_\zeta)^2]^{\frac{1}{2}}_{j,k} & \text{, along } \zeta\text{-direction} \quad (6.4) \end{cases}$$

$$L' = 1, \text{ and } \bar{W} = W(\xi,\zeta) \qquad (6.5)$$

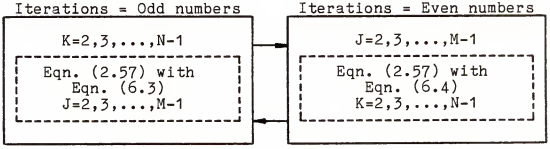
where $|D\xi|_{j,k}$ and $|D\zeta|_{j,k}$ are the magnitudes of the vectors, $\{D\xi\}$ and $\{D\zeta\}$, respectively, defined in Eqn. (2.12) at point (j,k) in the domain and on the boundary. Thus, L becomes the grid spacing $\Delta S_{j,k}$ and $\bar{W}$ represents the corresponding value of weight function $W_{j,k}$ in the computational space. Consequently, the parameters λ'_w and λ'_o in $\lambda_w \equiv \lambda/\lambda'_w$ and $\lambda_o \equiv \lambda/\lambda'_o$, respectively, are converted into variables given by

$$\left.\begin{aligned} (\lambda'_o)_{j,k} &= \lambda'_o(\xi,\zeta) = (\Delta S^4)_{j,k} \quad \text{, in } \Omega \\ (\lambda'_w)_{j,k} &= \lambda'_w(\xi,\zeta) = W_{j,k}(\Delta S^4)_{j,k} \quad \text{, in } \Omega \end{aligned}\right\} \quad (6.6)$$

$$\left.\begin{aligned} (\lambda'_w)_{j,1} &= \lambda'_w(\xi,\zeta) = W_{j,1}(\Delta S^3)_{j,1} \quad \text{, on inner } \Gamma \\ (\lambda'_w)_{j,N} &= \lambda'_w(\xi,\zeta) = W_{j,N}(\Delta S^3)_{j,N} \quad \text{, on outer } \Gamma \end{aligned}\right\} \quad (6.7)$$

The global parameter λ may be specified to depend on Δt, and will be assumed as $\lambda = 100\Delta t$ in an example. After substitution of Eqn. (6.7) and λ into Eqn. (2.59), the grid resolutions on the boundary are readily obtained by executing the Eqn. (2.64). Once the boundary conditions are

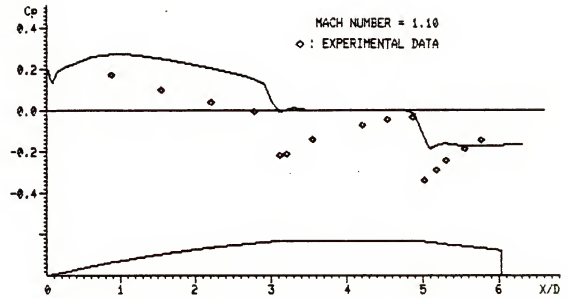
known, we then substitute Eqn. (6.6) and λ into (2.51) to obtain the grid resolutions within the domain by Eqn. (2.57) in alternating directions, such that



The adaptive grid generation based on the variation principles but implemented with a clustering strategy and variable penalty parameters λ_w' and λ_o' was investigated on the projectile flow problem at 1M = 0.91, 0.96 and 1.10. The local penalty parameters λ_w' and λ_o' were tested in those investigations with the expectation of obtaining good results. The strategy of this entire computation for the projectile inviscid flow problem is shown in Table 6.2, where NG is the number of changes of new grid system, DT is the time step Δt, λ is the global penalty parameter, NT is the number of time steps and DS is the clustering first grid spacing right next to the boundary surface. A sequence of the computed surface pressure coefficients C_p and the generated adaptive grid network for the case of free stream Mach number 1M = 1.10 is shown in Figs. 6.6-6.9. This sequence indicates clearly the process by which the grid network generated is adaptive extremely well to the pressure gradient field and exhibit the shock pattern. The computed

Table 6.2
Strategy of Implementation for Each NT = 150 Interval

NG	λ	DT	NT	DS
0		GENERATE THE ORIGINAL GRID		0.01
COMPUTE		0.005	1-50	
1	0.5	CHANGE TO THE NEW GRID		0.01
		0.005	51-100	
COMPUTE		0.010	101-150	
		0.010	151-200	
2	1.0	CHANGE TO THE NEW GRID		0.01
		0.025	201-250	
COMPUTE		0.025	251-300	
		0.050	301-350	
3	5.0	CHANGE TO THE NEW GRID		0.01
		0.050	351-400	
COMPUTE		0.075	401-450	
		0.075	451-500	
4	7.5	CHANGE TO THE NEW GRID		0.01
		0.100	501-550	
COMPUTE		0.100	551-600	
		0.100	601-650	
5	10.0	CHANGE TO THE NEW GRID		0.01
		0.100	651-700	
COMPUTE		0.100	701-750	
		0.100	751-800	



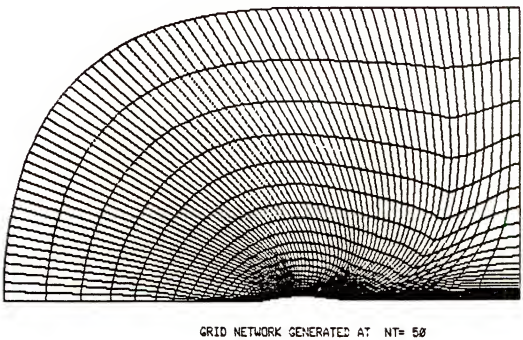
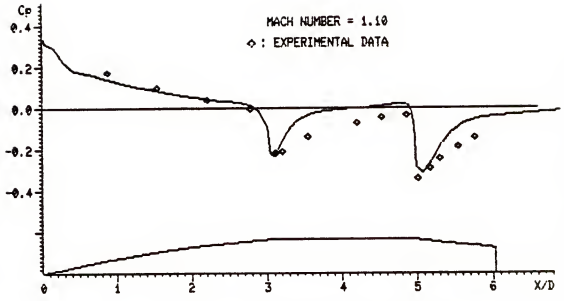


Fig. 6.6 Surface pressure and resulting grid network at NT = 50



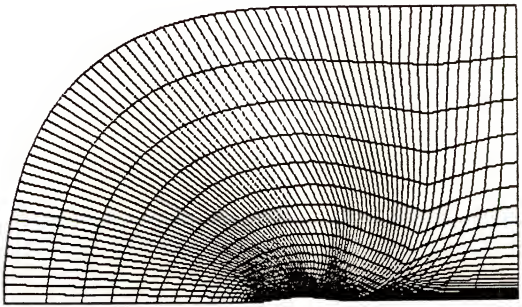
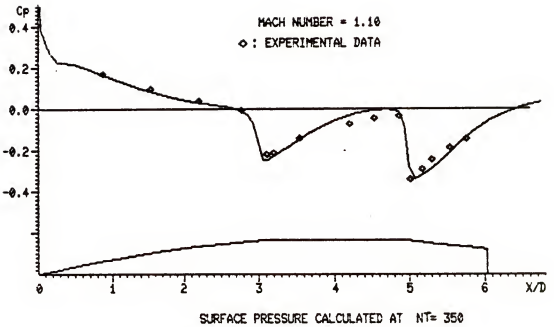


Fig. 6.7 Surface pressure and resulting grid network at NT = 200



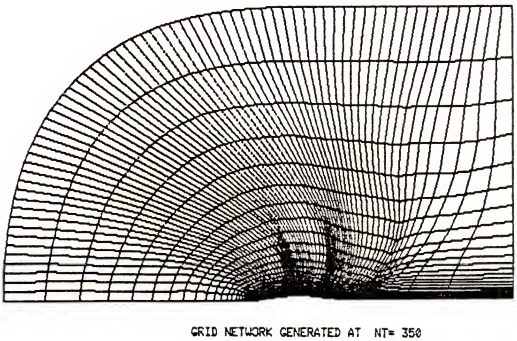
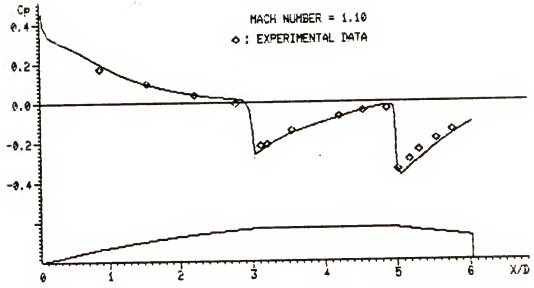


Fig. 6.8 Surface pressure and resulting grid network at NT = 350



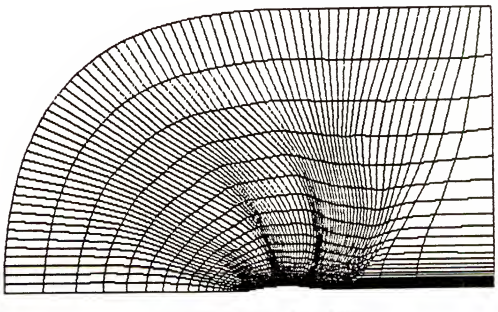


Fig. 6.9 Surface pressure and resulting grid network at NT = 500

surface pressure coefficient C_p, for 1M = 0.91, 0.96, and 1.10 obtained at NT = 650, 800, and 500, respectively, are shown in Fig. 6.10. Each of the three flow cases has been solved again to NT = 800 using a fixed but good adaptive grid system generated from the original GRIDGEN as described in Chapter V and the results obtained are shown in Fig. 6.11 for comparison. It is clear that the results obtained from adaptive gridding agree better with the experimental data.

Additional numerical experiments have been designed and conducted to gain some insight into the relative importance of adaptive boundary grids and internal grids to the accuracy and solution algorithm of the Navier-Stokes code. For each of the flow cases considered, a planar grid network is generated from the original GRIDGEN with the surface boundary grid distribution exactly the same as that of the corresponding adaptive grid network. Figure 6.12 shows the difference of the two grid systems for 1M = 1.10. The grid network generated is then employed in the Navier-Stokes code for the computation. Accurate results have been obtained for the case 1M = 1.10; however, for the other two cases 1M = 0.91 and 0.96, the solution algorithm of the Navier-Stokes code failed within NT = 30. Thus, for the fixed grid network, the original GRIDGEN may not generate a good interior grid network within the domain based on an adaptive boundary grid.

In order to find a better adaptive grid network further, there may be a better strategy implemented by

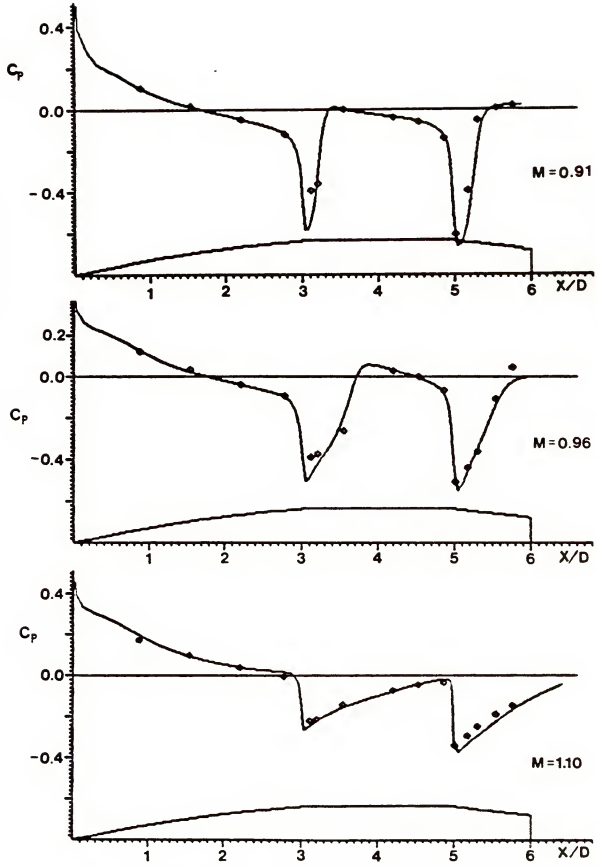


Fig. 6.10 Converged solutions (C_p) obtained from adaptive grid system

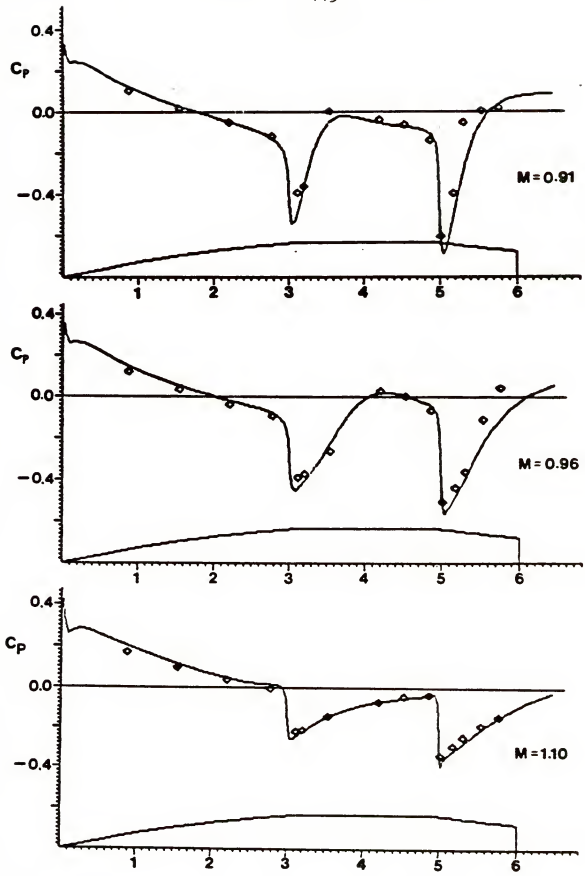
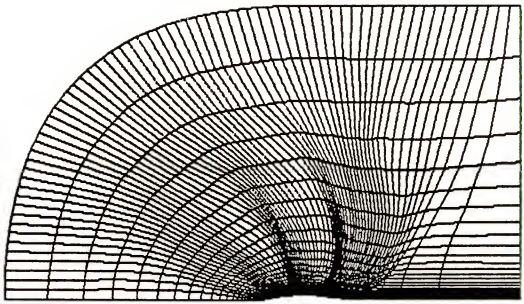
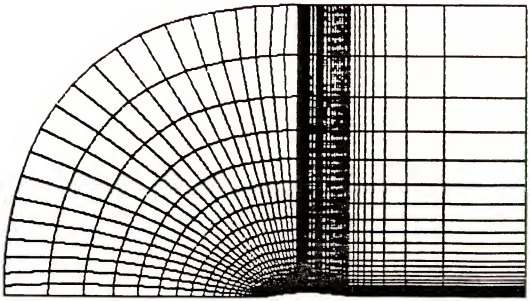


Fig. 6.11 Body surface pressure calculated on fixed grid network at NT = 800



(a) Adaptive grid network generated at NT = 500



(b) An adaptive grid system obtained from the original GRIDGEN with exactly the same surface boundary grids as those on Fig. 6.12(a)

Fig. 6.12 Adaptive grid networks generated for 1M = 1.10

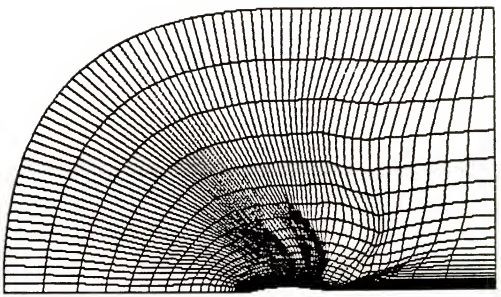
reducing the final time step from $\Delta t = 0.1$ to 0.5 and increasing λ in $\lambda_w = \lambda/\lambda'_w$ and $\lambda_o = \lambda/\lambda'_o$ gradually as described in Table 6.3. Figures 6.13 and 6.14 show the adaptive grid networks generated for each 100 time steps in the case 1M = 0.96 and the converged C_p-distribution calculated at NT = 1000. These results agree more smoothly and better with the experimental data than the results shown in Fig. 6.10 for 1M = 0.96. We then generate a planar grid network by the original GRIDGEN with the surface boundary grid distribution exactly the same as the corresponding adaptive grid network in Fig. 6.13; this fixed grid system is employed in the thin-layer Navier-Stokes code for computation and provides the results as shown in Fig. 6.15. It is clear that the results obtained from the adaptive grid in Fig. 6.13 agree better with the experimental data than the results obtained from the fixed grid in Fig. 6.15.

Also, from the results shown in Fig. 6.15, the calculated surface pressure C_p on the ogive and cylinder agrees well with the experimental data; however, the surface pressure is far away from the measured data between the boattail and sting. This is caused by a grid network being generated which is not adaptive to the solution. Thus, we conclude that satisfactory results depend strongly not only upon the adaptive grid points on the boundary but also upon the adaptive grids within the domain.

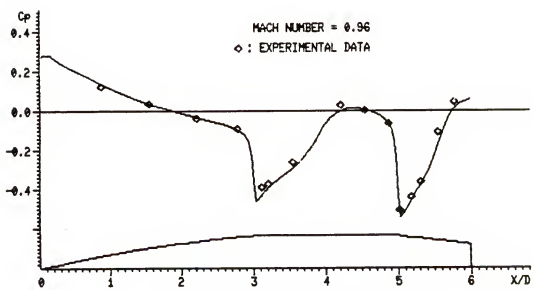
For summarizing the strategies of Tables 6.2 and 6.3, we also conclude that the global penalty parameters λ cannot

Table 6.3
Strategy of Implementation for Each NT = 100 Interval

NG	λ	DT	NT	DS
0		GENERATE THE ORIGINAL GRID		0.01
COMPUTE		0.005	1-100	
1	1.0	CHANGE TO THE NEW GRID		0.01
COMPUTE		0.010	101-200	
2	1.5	CHANGE TO THE NEW GRID		0.01
COMPUTE		0.025	201-300	
3	3.0	CHANGE TO THE NEW GRID		0.01
COMPUTE		0.050	301-400	
4	6.0	CHANGE TO THE NEW GRID		0.01
COMPUTE		0.050	401-500	
5	10.0	CHANGE TO THE NEW GRID		0.01
COMPUTE		0.050	501-600	
6	16.0	CHANGE TO THE NEW GRID		0.01
COMPUTE		0.050	601-700	
7	23.0	CHANGE TO THE NEW GRID		0.01
COMPUTE		0.050	701-800	
8	28.0	CHANGE TO THE NEW GRID		0.01
COMPUTE		0.050	801-900	
9	30.0	CHANGE TO THE NEW GRID		0.01
COMPUTE		0.050	901-1000	

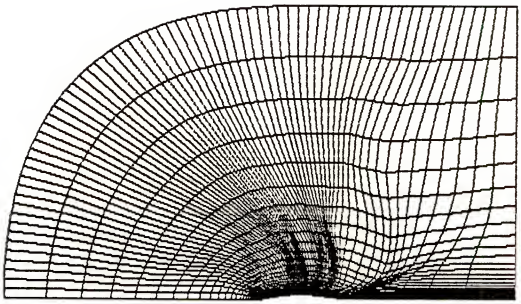


GRID NETWORK GENERATED AT NT= 900

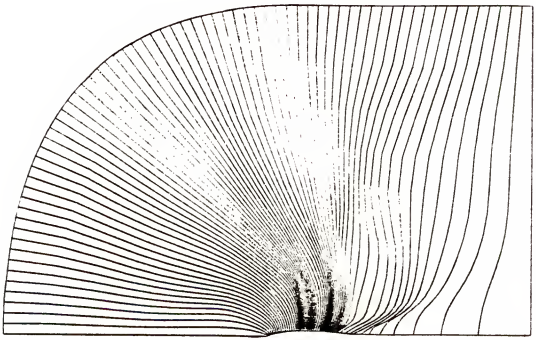


SURFACE PRESSURE CALCULATED AT NT= 1000

Fig. 6.13 Adaptive grid network and converged solution (C_p)

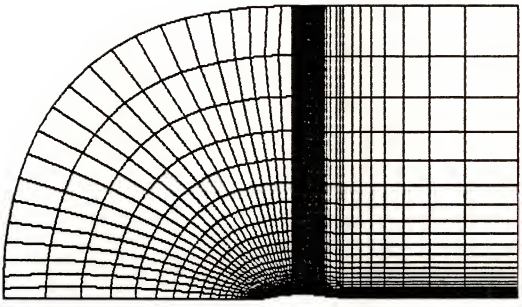


(a) Adaptive grid network generated at NT = 1000

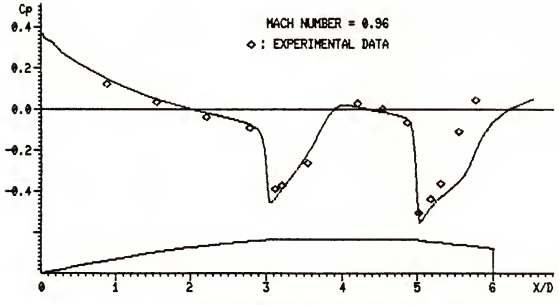


(b) The grid ζ-line exhibiting the shock patterns

Fig. 6.14 Grid network adaptive to the converged solution at NT = 1000



FIXED GRID NETWORK GENERATED BY "GRIDGEN"



SURFACE PRESSURE CALCULATED AT NT= 1000

Fig. 6.15 Adaptive grid system obtained from "GRIDGEN" with exactly the same surface boundary grids as those on Fig. 6.13 and its results calculated at NT = 1000

be used with a large number initially in the process of calculation; however, λ can be increased when the final time-step is fixed in the continuing computation. Therefore, the penalty parameters λ_w and λ_o can be certainly converted into variables which are not only in terms of the local position but also the time-step size Δt, such that

$$\begin{aligned} \lambda_w &= \lambda_w(\xi,\zeta,\Delta t) = \lambda(\Delta t)/\lambda'_w(\xi,\zeta) \\ \lambda_o &= \lambda_o(\xi,\zeta,\Delta t) = \lambda(\Delta t)/\lambda'_o(\xi,\zeta) \ . \end{aligned} \tag{6.9}$$

CHAPTER VII
CONCLUDING REMARKS

The adaptive grid generation based on the variational principle was investigated in detail for the projectile aerodynamics problem. The one-dimensional principle has been successfully applied to adaptive boundary grid generation, as well as the two-dimensional variational principle to the grid generation in the domain. Excellent consistency has been achieved with the same corresponding penalty parameters. The developed one-dimensional grid generator can be applied to any complex configuration of boundary such as a curvilinear surface combined with a polygonal shape.

The penalty parameters for orthogonality and concentration of grid, treated as constants in the work reported to date, have been investigated to show that they are of the same order of magnitude and can be treated as variable. These variables are in terms of not only the cell grid spacing over the entire domain but also the time-step size. The gradient of pressure is chosen as the control weight function in the computation of inviscid transonic flow problem. However, the generated grid spacing in the normal streamwise direction close to the body surface is

often too large and causes the numerical instability of solution algorithm. In order to solve this problem, the investigation has shown that an artificial clustering of the grid points close to the surface boundary can be used to ensure the convergent and stable computation in the Navier-Stokes code.

The adaptive grid generation code coupled with an axisymmetric version of the thin-layer Navier-Stokes code has been used successfully for the computation of the transonic projectile flow field. The coupled computer code offers an automatic calculation procedure for changing the time step, penalty parameters, and the adaptive grid network.

Comparisons were made of the results of computations from adaptive and fixed grid systems with experimental data for projectile inviscid transonic flow problem. The comparisons show that the adaptive grid system gives a better agreement with the experimental data than the fixed grid system.

APPENDIX A
NOMENCLATURE OF INPUT PARAMETERS IN THIN-LAYER CODE

A.1 NMAX, JMAX, KMAX, LMAX, ND

NMAX: The maximum number of time steps

JMAX: The maximum number of grid points along ξ-direction

KMAX: The maximum number of grid points along ϕ-direction

LMAX: The maximum number of grid points along ζ-direction

ND: The number usually equal to KMAX

A.2 DT, DX, DY, DZ, FSMACH, SMU, ALP

DT: Time step Δt

DX: Step size Δx of x-direction

DY: Step size Δy of y-direction

DZ: Step size Δz of z-direction

FSMACH: Free stream Mach number M_∞

SMU: Explicit smooth parameter ε_E

ALP: Angle of attack α

A.3 IREAD, IWRIT, NGRI, INVISC, NP

IREAD: Number of readings

0: starting from N=1 to N=NMAX

1: restarting from N=NMAX+1

IWRITE: Number of writings to control output

NGRI: Number of grid input data types

1: 2-dimensional

2: Tape 3

INVISC: Number of flow types

0: Inviscid flow

1: Viscous flow

NP: Number of time step intervals to print out the output

A.4 CNBR, OMEGA, PDOV

CNBR: Courrant number

OMEGA: Angular speed ω

PDOV: A factor transferring the Mach number to spin number (i.e., SPIN=PDOV*FSMACH)

A.5 RE, PR, RMUE, TZ

RE: Reynold's number Re

PR: Prandtl number Pr

RMUE: Coefficient of viscosity

TZ: Stagnation temperature T_o

A.6 LAMIN, RM

LAMIN: Number of controls of flow case

0: Laminar flow

1: Turbulent flow

RM: Implicit smooth parameter ε_I

APPENDIX B
PARTIAL PROGRAM LISTING

The grid generation code "GRIDGEN" can be implemented by adding several subprograms so that the user will have an option to generate an adaptive grid network for the accurate solutions. The implementation and modification are described in this Appendix. In the process, efforts have been made to keep the changes as minimal as possible. In fact, only seven subroutines (ADGRD,GRADP,WFCN1,ADBD,WFCN2, DIFF, and RELAX) with executable statements are added to GRIDGEN to generate adaptive grid points on the boundary and in the domain. The modifications made to GRIDGEN and the relevant information required for using adaptive GRIDGEN are described in the following:

A. Input Data

Two data cards are added at the beginning of the input data file of GRIDGEN, and their executable statements now read these two additional input data cards as Hsu [8] used the statements in the program MAIN of GRIDGEN. The first datum read is IADAPT, such that

$$\text{IADAPT} \begin{cases} 0 \text{ ; implemented by GRIDGEN} \\ 1 \text{ ; implemented by adaptive GRIDGEN} \end{cases}$$

The second data read are

IE(1),IE(2),IE(3),IE(4),IE(5),IE(6),IE(7),IE(8)

These data provide information and instruction for the change of number of grid points on each segment of inner boundary (i.e., the change of boundary grid spacing). For example, there are 23 points on ogive, 22 points on the cylinder, 17 points on the boattail, and 8 points on the sting. We then set IE(1)=1, IE(2)=23, IE(3)=23, IE(4)=45, IE(5)=45, IE(6)=62, IE(7)=62, and IE(8)=70. The use of the COMMON IADAPT and COMMON IE(8) are available for the relevant subprograms such as they are used in ADGRD to generate the adaptive grid points on each segment of boundary. The program then continues as before to generate inner boundary grid points by calling appropriate subroutines.

B. Inserted Statements

In order to add a new option for adaptive grid generation into GRIDGEN. It is suitable to insert two statements in the subroutine "ALGRD," the first one is the "IF" statement for the optional number IADAPT=1, then we go to the second statement "CALL ADGRD," otherwise we bypass this executable statement. Since we can use directly the implementation of clustering grid points close to the boundary after the adaptive grid generated in the subroutine ALGRD of GRIDGEN, this is why we insert the execution statement for adaptive grid generation initially in ALGRD.

C. Inserted Subprograms

ADGRD:

This subroutine is an adaptive grid solver called when IADAPT=1, the subroutines GRADP, ADBD, and RELAX are called.

GRADP:

The calculations of pressure gradients or any other gradients with one dimension only for the boundary will be executed in preparation for constructing the weight function. Also, the calculated pressure gradients are modified by applying Eqn. (5.4).

WFCN1(---):

The one-dimensional boundary weight function is executed by applying Eqn. (5.2) with the modifying Eqn. (5.6).

ADBD(---):

This subroutine is the calculation of adaptive boundary grid points executed by Eqn. (2.64). Prior to implementation of the Eqn. (2.64), the Cartesian coordinates are transformed into curvilinear coordinates, thus, the boundary points may be described in one dimension. We then apply the Newton-Raphson method with SOR to calculate the adaptive grid on the boundary. Also, we have to transfer the curvilinear system back to the Cartesian system by employing cubic spline function as the interpolating function to find

new (x,z) values on the boundary. This subroutine is called by ADGRD five times (i.e., an outer boundary plus four segments combined to an inner boundary of projectile surface). The subroutine WFCN1 is called.

WFCN2(-):

The two-dimensional weight function is calculated by Eqn. (5.1) with the modifying Eqn. (5.4) and (5.5). Also, the Jacobian determinant is calculated after each iteration in RELAX. The derivatives of weight function, W_x and W_z, with transformations are also calculated.

DIFF(---):

This subprogram calculates the metrics, second derivatives Jacobian determinant, coefficients of highest derivative terms and residuals of the coupled Eqn. (2.36).

RELAX(---):

This subprogram is the solver for two-dimensional grid calculations executed by the N-R method with SOR in using Eqn. (2.57) and (2.58). The subroutine WFCN2 is called for each iteration since the Jacobian determinant should be changed currently. The subroutine DIFF is called to calculate either the adaptive grid points if IADAPT=1 or regular grid points if IADAPT=0. The last task is to extrapolate linearly the interior grid lines orthogonal to the axis and downstream boundary. The iteration continues until it converges up to either the

allowable control residual RES=0.01 or the specified maximum iteration number ITERM=500. Note that this new subroutine RELAX is replaced by the old "RELAX" in GRIDGEN.

D. The Flow Program

The summarization of above descriptions results in the flow program for adaptive grid calculation as shown in Fig. A.1.

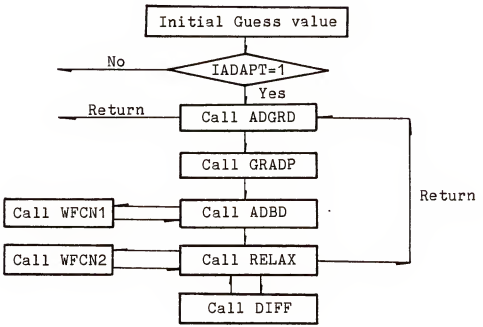


Fig. A.1 Flow program for adaptive grid calculation

E. Computer Subprograms

The computer subprograms are shown in the following pages.

```
      SUBROUTINE ADGRD
      IMPLICIT DOUBLE PRECISION *6(A-H,O-Z)
      DOUBLE PRECISION *6 LBO,LBV
      COMMON JMAX, KMAX, JM, KM, NBOD, JBOD
      COMMON/GRD/X1(90,100),Y1(90,100)
      COMMON/VARS/Q(90,6,100)
      COMMON/VAR3/P(60,100)
      COMMON /BOUDY/ XX(100), YY(100), XS(100), YS(100), SS(100), S(100)
     1  , T(100), TS(100)
      COMMON /OUTER/ JA(8), JB(8)
      COMMON /C/ CA0(8), CA1(8), CA2(8), CB0(8), CB1(8), CB2(8), CARC(8)
      COMMON /ARRAY/ A(100), B(100), C(100), D(100), F(100), H(100)
      COMMON /FIX/ XFIX, YFIX
      COMMON /ELLIP/ RXXD,RXYD,RYXD,RYYD,RX,RY,RDET,LBO,LBV,LBS
      COMMON/PYHSU/IADAPT
      COMMON/COUNT2/NN,ISCAL
      COMMON /PARA/ AFA,BTA
      COMMON/WEIGHT/W(90,100),DWX(90,100),DWY(90,100)
      COMMON/SEGP/IE(8)
      COMMON/GGONLY/IGG,NGG,NGMAX,MM
      AFA=1.E-6
      BTA=1.0
      EPS=1.D-4
C
C     CALL OUTPUT TO FIND THE PRECEDING APPROXIMATE SOLUTION
C
      CALL OUTPUT(0,0,0,ISCAL,1)
C
C     FIND THE PRESSURE GRADIENTS ALONG THE BOUNDARY
C
      CALL GRADP
C
C     CALCULATE THE ADAPTIVE GRID ON INNER BOUNDARY
C     NOTE THAT THERE ARE FOUR SEGMENTS COMBINED THUS WE MUST
C     CALCULATE SEPARATELY
C
      N1=IE(1)+1
      N2=IE(2)-1
      CALL ADBD(ITERM,EPS,WMEGA,N1,N2,1,1,1,1,1)
      DO 43 J=N1,N2
      X1(J,1)=T(J)
      Y1(J,1)=TS(J)
   43 CONTINUE
      N3=IE(3)+1
      N4=IE(4)-1
      CALL ADBD(ITERM,EPS,WMEGA,N3,N4,1,1,1,1,1)
      DO 44 J=N3,N4
   44 X1(J,1)=T(J)
      N5=IE(5)+1
      N6=IE(6)-1
      CALL ADBD(ITERM,EPS,WMEGA,N5,N6,1,1,1,1,1)
      DO 46 J=N5,N6
      X1(J,1)=T(J)
      Y1(J,1)=TS(J)
   46 CONTINUE
```

```
      N7=IE(7)+1
      CALL ADBD(ITERM,EPS,WMEGA,N7,JM,1,1,1,1,1)
      DO 48 J=N7,JM
      X1(J,1)=T(J)
   48 CONTINUE
C
C     CALCULATE THE ADAPTIVE GRID ON OUTER BOUNDARY
C
      NJ=JB(1)
      NP=NJ+1
      NM=NJ-1
      EPM=1.D-2
      CALL ADBD(ITERM,EPM,WMEGA,2,JM,1,KMAX,KMAX,1,1)
      DO 54 J=2,JM
      X1(J,KMAX)=T(J)
      Y1(J,KMAX)=TS(J)
   54 CONTINUE
C
C     CALCULATE THE ADAPTIVE GRID IN THE DOMAIN
C
      RESD=1.D-2
      CALL RELAX(ITERM,RESD,WMEGA,2,JM,1,2,KM,1)
      WRITE (16,200) ((X1(J,K),J=1,JMAX),K=1,KMAX),((Y1(J,K),J=1,JMAX),
     1          K=1,KMAX)
      IF(MM .GE. 2) NCLUS=3
      RETURN
  200 FORMAT(14F9.5)
      END
```

```
      SUBROUTINE GRADP
      IMPLICIT DOUBLE PRECISION *6(A-H,O-Z)
      COMMON JMAX, KMAX, JM, KM, NBOD, JBOD
      COMMON/VAR3/P(60,100)
      COMMON/VARS/Q(90,6,100)
      COMMON /GRAD/ DPDX(60,100),DPDY(60,100)
      COMMON/GANDM/RGAMA,RMACH
C
C     CALCULATE THE PRESSURE GRADIENTS
C
      DO 20 K=1,KMAX,KM
      DO 10 J=2,JM
      DPDX(K,J)=0.5*(P(K,J+1)-P(K,J-1))
   10 CONTINUE
      DPDX(K,1)=0.5*(-3.*P(K,1)+4.*P(K,2)-P(K,3))
      DPDX(K,JMAX)=0.5*(3.*P(K,JMAX)-4.*P(K,JM)+P(K,JM-1))
   20 CONTINUE
C
C     TAKE AVERAGE AT THE CHANGED SIGN OF PRESSURE GRADIENTS
C
      DO 30 K=1,KMAX,KM
      DO 30 J=2,JM
      JP=J+1
      JR=J-1
      PXR=P(K,J)-P(K,JR)
      PXP=P(K,JP)-P(K,J)
      XPR=PXP*PXR
      IF(XPR .LT. 0.0) DPDX(K,J)=0.5*(DPDX(K,JP)-DPDX(K,JR))
   30 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE WFCN1(ITER,J1,J2,JV,K1,K2,KV,JO)
      IMPLICIT DOUBLE PRECISION *6(A-H,O-Z)
      COMMON JMAX, KMAX, JM, KM, NBOO, JBOO
      COMMON/GRD/X1(90,100),Y1(90,100)
      COMMON/VAR3/P(60,100)
      COMMON/VARS/Q(90,6,100)
      COMMON/WEIGHT/W(90,100),DWX(90,100),DWY(90,100)
      COMMON /ELLIP/ RXXO,RXYO,RYXO,RYYO,RX,RY,RDET,LBO,LBV,LBS
      COMMON /ARRAY/ A(100), B(100), C(100), D(100), F(100), H(100)
      COMMON /BOUOY/ XX(100), YY(100), XS(100), YS(100), SS(100), S(100)
     1  , T(100), TS(100)
      COMMON /GRAD/ DPDX(60,100),DPOY(60,100)
      COMMON /PARA/ AFA,BTA
      IMAX=3
      K=K1
      JI=J1-1
      JF=J2+1
C
C     TRANSFER TO CURVILINEAR SYSTEM FOR ONE-DIMENSION ON BOUNDARY
C
      IF(ITER .GT. 1) GO TO 40
      SS(JI)=0.0
      DO 40 J=J1,JF,JV
      JR=J-1
      SS(J)=SS(JR)+DSQRT((X1(J,K)-X1(JR,K))**2+(Y1(J,K)-Y1(JR,K))**2)
   40 CONTINUE
C
C     TRANSFER THE PRESSURE GRADIENT TO WEIGHT FUNCTION
      DO 60 J=J1,J2,JV
      SXD=0.5*(SS(J+1)-SS(J-1))
      W(K,J)=DPDX(K,J)/SXD
      W(K,J)=AFA+BTA*DABS(W(K,J))
   60 CONTINUE
      SX01=0.5*(-3.*SS(JI)+4.*SS(J1)-SS(J1+1))
      SXDM=0.5*(3.*SS(JF)-4.*SS(J2)+SS(J2-1))
      W(K,JI)=DPDX(K,JI)/SXD1
      W(K,JF)=DPDX(K,JF)/SXDM
      W(K,JI)=AFA+BTA*DABS(W(K,JI))
      W(K,JF)=AFA+BTA*DABS(W(K,JF))
C
C     CALCULATE THE WEIGHT FUNCTION SMOOTHLY
      IF(K .EQ. KMAX) IMAX=20
      DO 85 I=1,IMAX
      DO 80 J=J1,J2,JV
      WW=0.5*(W(K,J+1)+W(K,J-1))
      W(K,J)=W(K,J)+0.4*(WW-W(K,J))
   80 CONTINUE
   85 CONTINUE
C
C     CALCULATE THE DERIVATIVE OF WEIGHT FUNCTION
      DO 90 J=J1,J2,JV
      DWX(K,J)=0.5*(W(K,J+1)-W(K,J-1))
   90 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE  ADBD(ITERM,RES,WMEGA,J1,J2,JV,K1,K2,KV,JO)
      IMPLICIT DOUBLE PRECISION *6(A-H,O-Z)
      DOUBLE PRECISION *6 JCOB,LAX,LBY,LBX,LCY,L3O,L3V,LBS
      COMMON JMAX, KMAX, JM, KM, N80D, J80D
      COMMON/GRO/X1(90,100),Y1(90,100)
      COMMON/VARS/Q(90,6,100)
      COMMON/VAR3/P(60,100)
      COMMON/WEIGHT/W(90,100),DWX(90,100),DWY(90,100)
      COMMON /ELLIP/ RXXD,RXYD,RYXD,RYYD,RX,RY,RDET,LBO,LBV,LBS
      COMMON /ARRAY/ A(100), B(100), C(100), D(100), F(100), H(100)
      COMMON /BOUDY/ XX(100), YY(100), XS(100), YS(100), SS(100), S(100)
     1  , T(100), TS(100)
      COMMON /PARA/ AFA,BTA
C
C     CALCULATION FOR ADAPTIVE BOUNDARY GRID POINTS BY USING
C     NEWTON-RAPHSON ITERATION WITH SUCCESSIVE OVER RELAXATION METHOD
C
C     BEGIN TO ITERATE
      DO 60 ITER=1,ITERM
      ITCON = 1
      CALL WFCN1(ITER,J1,J2,JV,K1,K2,KV,JO)
      K=K1
      JI=J1-1
      JF=J2+1
C
C     NORMALIZE THE LUMBDA W BY DIMENSION ANALYSIS
C
      WT=0.0
      DO 20 J=J1,JF
   20 WT=WT+0.5*(W(K,J)+W(K,J-1))*(SS(J)-SS(J-1))
      TP=JF-JI
      TL=SS(JF)
      SL=(TL/TP)**3
      WBAR=WT/TL
      SLBV=WBAR*SL
      RBV=LBV/SLBV
      IF(RBV .GT. 1.D6) RBV=1.D6
C
C     CALCULATE THE ADAPTIVE BOUNDARY POINTS
C
      DO 40 J=J1,J2,JV
      JP=J+1
      JR=J-1
      SXD=0.5*(SS(JP)-SS(JR))
      SXXD=SS(JP)-2.*SS(J)+SS(JR)
      RS=SXXD+RBV*(SXD**4)*DWX(K,J)/(2.*(1.+RBV*(SXD**3)*W(K,J)))
      RSXD=-2.0
      RRS=RS/RSXD
      SS(J)=SS(J)-WMEGA*RRS
C     CONTROL THE ACCURACY OF SOLUTION
      ARS=DABS(RS)
      IF(ARS .GT. RES) ITCON = 0
   40 CONTINUE
      IF(ITCON.NE.0) GO TO 80
   60 CONTINUE
```

```
   80 CONTINUE
C
C     TRANSFER BACK TO CARTESIAN SYSTEM BY USING CUBIC SPLINE
C     INTERPOLATION
C
      IF(K .EQ. KMAX) GO TO 120
C
C     CALCULATE THE X,Y VALUES ON INNER BOUNDARY
C
      S(JI)=0.0
      DO 100 J=J1,JF
  100 S(J)=S(J-1)+DSQRT((XX(J)-XX(J-1))**2+(YY(J)-YY(J-1))**2)
      CALL CSPLIN(SS,T,S,XX,A,B,C,D,F,H,J1,J2,JI,JF)
      CALL CSPLIN(SS,TS,S,YY,A,B,C,D,F,H,J1,J2,JI,JF)
      GO TO 160
C
C     CALCULATE THE X,Y VALUES ON OUTER BOUNDARY
C
  120 CONTINUE
      S(JI)=0.0
      DO 140 J=J1,JF
      JR=J-1
      S(J)=S(JR)+DSQRT((XS(J)-XS(JR))**2+(YS(J)-YS(JR))**2)
  140 IF(S(J) .EQ. S(JR)) S(JR)=0.5*(S(J)+S(JR-1))
      CALL CSPLIN(SS,T,S,XS,A,B,C,D,F,H,J1,J2,JI,JF)
      CALL CSPLIN(SS,TS,S,YS,A,B,C,D,F,H,J1,J2,JI,JF)
  160 CONTINUE
      TL=SS(JF)
      WRITE(6,200) ITER,TL
C     WRITE(6,190) (W(K,J),J=JI,JF)
  190 FORMAT(12E11.4)
  200 FORMAT(//5X,"THE NUMBER OF ITERATIONS = ",I4,6X,
     1 "TOTAL LENGTH = ",E11.4/)
      RETURN
      END
```

```
      SUBROUTINE RELAX(ITERM,RES,WMEGA,J1,J2,JV,K1,K2,KV)
      IMPLICIT DOUBLE PRECISION *6(A-H,O-Z)
      DOUBLE PRECISION *6 JCOB,LAX,LBY,LBX,LCY,LBO,LBV,LBS
      COMMON JMAX, KMAX, JM, KM, NBOD, JBOD
      COMMON/GRD/X1(90,100),Y1(90,100)
      COMMON/VARS/Q(90,6,100)
      COMMON/VARI/P(90,100)
      COMMON/WEIGHT/W(90,100),DWX(90,100),DWY(90,100)
      COMMON /BLOCK2/ XXF(80,40),XEF(80,40),YXF(80,40),YEF(80,40)
      COMMON /ELLIP/ RXXD,RXYD,RYXD,RYYD,RX,RY,RDET,LBO,LBV,LBS
      COMMON/PYHSU/IADAPT
      COMMON/COUNT2/NN,ISCAL
      COMMON /ARRAY/ A(100), B(100), C(100), D(100), F(100), H(100)
      COMMON /BOUDY/ XX(100), YY(100), XS(100), YS(100), SS(100), S(100)
     1  , T(100), TS(100)
      COMMON /PARA/ AFA,BTA
C
C     NEWTON-RAPHSON ITERATION WITH SUCCESSIVE OVER RELAXATION METHOD
C
      XO=X1(1,1)
      JI=J1-1
      JF=J2+1
      KI=K1-1
      KF=K2+1
      IF(IADAPT .EQ. 0) RBO=1.0
      IF(IADAPT .EQ. 0) RBV=0.0
C     BEGIN TO ITERATE BY USING ADI METHOD
      DO 90 ITER=1,ITERM
      ITCON=1
      IF(IADAPT .EQ. 0) GO TO 10
      CALL WFCN2 (ITER)
   10 CONTINUE
      IF(MOD(ITER,2) .EQ. 0) GO TO 45
      DO 40 K=K1,K2,KV
C
C      ITERATING CALCULATION ALONG THE XI-LINE
C
      DO 40 J=J1,J2,JV
C
C     NORMALIZE THE LUMBDA V AND LUMBDA C BY DIMENSION ANALYSIS
      IF(IADAPT .EQ. 0) GO TO 35
      DSX=DSQRT((XXF(J,K))**2+(YXF(J,K))**2)
      SL=DSX**4
      SLBV=W(K,J)*SL
      RBV=LBV/SLBV
      RBO=LBO/SL
   35 CONTINUE
      CALL DIFF(J,K,RBO,RBV)
      RRX=(RX*RYYD-RY*RXYD)/RDET
      RRY=(RY*RXXD-RX*RYXD)/RDET
      X1(J,K)=X1(J,K)+WMEGA*RRX
      Y1(J,K)=Y1(J,K)+WMEGA*RRY
C     CONTROL THE ACCURACY OF SOLUTION
      ARX=DABS(RX)
      ARY=DABS(RY)
```

```
      IF(ARX.GT.RES.AND.ARY.GT.RES) ITCON=0
   40 CONTINUE
      GO TO 70
   45 CONTINUE
      DO 70 J=J1,J2,JV
C
C      ITERATING CALCULATION ALONG THE ETA-LINE
C
      DO 70 K=K1,K2,KV
C
C     NORMALIZE THE LUMBDA V AND LUMBDA O BY DIMENSION ANALYSIS
      IF(IADAPT .EQ. 0) GO TO 60
      DSE=DSQRT((XEF(J,K))**2+(YEF(J,K))**2)
      SL=DSE**4
      SLBV=W(K,J)*SL
      RBV=LBV/SLBV
      RBO=LBO/SL
   60 CONTINUE
      CALL DIFF(J,K,RBO,RBV)
      RRX=(RX*RYYD-RY*RXYD)/RDET
      RRY=(RY*RXXD-RX*RYXD)/RDET
      X1(J,K)=X1(J,K)+WMEGA*RRX
      Y1(J,K)=Y1(J,K)+WMEGA*RRY
C     CONTROL THE ACCURACY OF SOLUTION
      ARX=ABS(RX)
      ARY=ABS(RY)
      IF(ARX.GT.RES.AND.ARY.GT.RES) ITCON=0
   70 CONTINUE
      IF(ITCON.NE.0) GO TO 100
      KP=1
      XT=X1(2,2)
      IF(XT .LT. XO) GO TO 95
   75 KP=KP+1
      XT=X1(2,KP)
      IF(XT .LT. XO) GO TO 80
      GO TO 75
   80 CONTINUE
      TX1=X1(2,KP)-XO
      TX2=X1(2,KP)-X1(2,1)
      DO 85 K=2,KP
      DX2=X1(2,K)-X1(2,K-1)
      X1(1,K)=X1(1,K-1)+TX1*(DX2/TX2)
      Y1(JMAX,K)=Y1(JM,K)
   85 CONTINUE
      KP=KP+1
      DO 90 K=KP,KM
      X1(1,K)=X1(2,K)
      Y1(JMAX,K)=Y1(JM,K)
   90 CONTINUE
  100 CONTINUE
      WRITE(6,200) ITER,LBO,LBV
  200 FORMAT(//5X,"THE NUMBER OF ITERATIONS = ",I4/6X,
     1 "LUMB O = ",E11.4,5X,"LUMB V = ",E11.4//)
      RETURN
      END
```

```
      SUBROUTINE DIFF(J,K,RBO,RBV)
      IMPLICIT DOUBLE PRECISION *6(A-H,O-Z)
      DOUBLE PRECISION *6 JCOB,LAX,LBY,LBX,LCY,LBO,LBV,LBS
      COMMON JMAX, KMAX, JM, KM, NBOO, JBOO
      COMMON/GRD/X1(90,100),Y1(90,100)
      COMMON/VARS/Q(90,6,100)
      COMMON/VAR3/P(60,100)
      COMMON/WEIGHT/W(90,100),DWX(90,100),DWY(90,100)
      COMMON /BLOCKD/ XXF(80,40),XEF(80,40),YXF(80,40),YEF(80,40)
      COMMON /ELLIP/ RXXD,RXYD,RYXD,RYYD,RX,RY,RDET,LBO,LBV,LBS
      COMMON/PYHSU/IADAPT
      COMMON /PARA/ AFA,BTA
      JP=J+1
      JR=J-1
      KP=K+1
      KR=K-1
C
C     CALCULATE THE FIRST DERIVATIVE TERMS
C
      XXD=(X1(JP,K)-X1(JR,K))/2.0
      XED=(X1(J,KP)-X1(J,KR))/2.0
      YXD=(Y1(JP,K)-Y1(JR,K))/2.0
      YED=(Y1(J,KP)-Y1(J,KR))/2.0
C
C     CALCULATE THE SECOND DERIVATIVE TERMS
C
      XXXD=X1(JP,K)-2.*X1(J,K)+X1(JR,K)
      XEED=X1(J,KP)-2.*X1(J,K)+X1(J,KR)
      YXXD=Y1(JP,K)-2.*Y1(J,K)+Y1(JR,K)
      YEED=Y1(J,KP)-2.*Y1(J,K)+Y1(J,KR)
      XXED=(X1(JP,KP)-X1(JP,KR)-X1(JR,KP)+X1(JR,KR))/4.0
      YXED=(Y1(JP,KP)-Y1(JP,KR)-Y1(JR,KP)+Y1(JR,KR))/4.0
C
C     CALCULATE THE JACOBIAN DETERMINANT
C
      JCOB=XXD*YED-XED*YXD
      IF(JCOB.EQ.0.0) GO TO 140
C
C     CALCULATE THE VALUES OF SOURCE TERMS OF THE EQUATION
C     THE DECISION OF THE ADAPTIVE GRID GENERATION
C
      IF(IADAPT .EQ. 0) GO TO 40
      SOX=0.5*JCOB*DWX(K,J)
      SOY=0.5*JCOB*DWY(K,J)
   40 CONTINUE
C
C      CALCULATE THE COEFFICIENTS OF HIGHEST DERIVATIVE TERMS
C
      ALFA=XED**2+YED**2
      BETA=-2.*(XXD*XED+YXD*YED)
      GAMA=XXD**2+YXD**2
      AA=(YXD**2+YED**2)/(JCOB**3)
      BB=-(XXD*YXD+XED*YED)/(JCOB**3)
      CC=(XXD**2+XED**2)/(JCOB**3)
      CAA=AA*ALFA+RBO*XED*XED+RBV*YED*YED*W(K,J)
```

```
      CAB=AA*BETA+R80*2.*(2.*XXD*XED+YXD*YED)-R8V*(2.*YXD*YED)*W(K,J)
      CAC=AA*GAMA+R80*XXD*XXD+R8V*YXD*YXD*W(K,J)
      CBA=BB*ALFA+R80*XED*YED-R8V*XED*YED*W(K,J)
      CBB=BB*BETA+R80*(XXD*YED+XED*YXD)+R8V*(XXD*YED+XED*YXD)*W(K,J)
      CBC=BB*GAMA+R80*XXD*YXD-R8V*XXD*YXD*W(K,J)
      CCA=CC*ALFA+R80*YED*YED+R8V*XED*XED*W(K,J)
      CCB=CC*BETA+R80*2.*(XXD*XED+2.*YXD*YED)-R8V*(2.*XXD*XED)*W(K,J)
      CCC=CC*GAMA+R80*YXD*YXD+R8V*XXD*XXD*W(K,J)
      LAX=CAA*XXXD+CAB*XXED+CAC*XEED
      LBY=CBA*YXXD+CBB*YXED+CBC*YEED
      LBX=CBA*XXXD+CBB*XXED+CBC*XEED
      LCY=CCA*YXXD+CCB*YXED+CCC*YEED
C
C     CALCULATE THE RESIDUALS OF THE COUPLED EQUATIONS
C
      RX=LAX+LBY*R8V*SOX
      RY=LBX+LCY+R8V*SOY
      RXXD=-2.*(CAA+CAC)
      RXYD=-2.*(CBA+CBC)
      RYXD=-2.*(CBA+CBC)
      RYYD=-2.*(CCA+CCC)
      RDET=RXYD*RYXD-RXXD*RYYD
  120 CONTINUE
      RETURN
  140 WRITE(6,180)
      STOP
  180 FORMAT(///5X,"JACOBI DETERMINANT IS SINGULAR")
      END
```

```
      SUBROUTINE WFCN2 (ITER)
      IMPLICIT DOUBLE PRECISION *6(A-H,O-Z)
      COMMON JMAX, KMAX, JM, KM, N80D, J80D
      COMMON/GRD/X1(90,100),Y1(90,100)
      COMMON/VAR3/P(60,100)
      COMMON/VARS/Q(90,6,100)
      COMMON/WEIGHT/W(90,100),DWX(90,100),DWY(90,100)
      COMMON /BLOCKD/ XXF(80,40),XEF(80,40),YXF(80,40),YEF(80,40)
      COMMON /ELLIP/ RXXD,RXYD,RYXD,RYYD,RX,RY,RDET,LBO,LBV,LBS
      COMMON /ARRAY/ A(100), B(100), C(100), D(100), F(100), H(100)
      COMMON /BOUDY/ XX(100), YY(100), XS(100), YS(100), SS(100), S(100)
     1   , T(100), TS(100)
      COMMON /GRAD/ DPDX(60,100),DPDY(60,100)
      COMMON /PARA/ AFA,BTA
      COMMON/GANDM/RGAMA,RMACH
      IMAX=3
C
C     CALCULATE THE METRICS AND JACOBIAN AT EACH ITERATION FROM RELAX
C
      KV=1
      DO 20 K=1,KMAX,KV
      DO 10 J=2,JM
      XXF(J,K)=(X1(J+1,K)-X1(J-1,K))/2.0
      YXF(J,K)=(Y1(J+1,K)-Y1(J-1,K))/2.0
   10 CONTINUE
      XXF(1,K)=(-3.*X1(1,K)+4.*X1(2,K)-X1(3,K))/2.0
      YXF(1,K)=(-3.*Y1(1,K)+4.*Y1(2,K)-Y1(3,K))/2.0
      XXF(JMAX,K)=(3.*X1(JMAX,K)-4.*X1(JM,K)+X1(JM-1,K))/2.0
      YXF(JMAX,K)=(3.*Y1(JMAX,K)-4.*Y1(JM,K)+Y1(JM-1,K))/2.0
   20 CONTINUE
      JV=1
      DO 40 J=1,JMAX,JV
      DO 30 K=2,KM
      XEF(J,K)=(X1(J,K+1)-X1(J,K-1))/2.0
      YEF(J,K)=(Y1(J,K+1)-Y1(J,K-1))/2.0
   30 CONTINUE
      XEF(J,1)=(-3.*X1(J,1)+4.*X1(J,2)-X1(J,3))/2.0
      YEF(J,1)=(-3.*Y1(J,1)+4.*Y1(J,2)-Y1(J,3))/2.0
      XEF(J,KMAX)=(3.*X1(J,KMAX)-4.*X1(J,KM)+X1(J,KM-1))/2.0
      YEF(J,KMAX)=(3.*Y1(J,KMAX)-4.*X1(J,KM)+Y1(J,KM-1))/2.0
   40 CONTINUE
      DO 50 K=1,KMAX,KV
      DO 50 J=1,JMAX
      DINV=XXF(J,K)*YEF(J,K)-XEF(J,K)*YXF(J,K)
      IF(DINV .EQ. 0.0) GO TO 50
      Q(K,6,J) = 1./DINV
   50 CONTINUE
C
C     CALCULATE THE PRESSURE GRADIENTS OVERALL FLOW FIELD
C
      IF(ITER .GT. 1) GO TO 120
      DO 80 K=1,KMAX
      DO 70 J=2,JM
      DPDX(K,J)=0.5*(P(K,J+1)-P(K,J-1))
   70 CONTINUE
```

```
      DPDX(K,1)=0.5*(-3.*P(K,1)+4.*P(K,2)-P(K,3))
      DPDX(K,JMAX)=0.5*(3.*P(K,JMAX)-4.*P(K,JM)+P(K,JM-1))
   80 CONTINUE
      DO 90 K=1,KM1X
      DO 90 J=2,JM
      PXR=P(K,J)-P(K,J-1)
      PXP=P(K,J+1)-P(K,J)
      XPR=PXP*PXR
      IF(XPR .LT. 0.0) DPDX(K,J)=0.5*(DPDX(K,J+1)-DPDX(K,J-1))
C     IF(XPR .LT. 0.0) DPDX(K,J)=0.5*(PXP-PXR)
   90 CONTINUE
      DO 110 J=1,JMAX
      DO 100 K=2,KM
      DPDY(K,J)=0.5*(P(K+1,J)-P(K-1,J))
  100 CONTINUE
      DPDY(1,J)=0.5*(-3.*P(1,J)+4.*P(2,J)-P(3,J))
      DPDY(KMAX,J)=0.5*(3.*P(KMAX,J)-4.*P(KM,J)+P(KM-1,J))
  110 CONTINUE
      DO 120 J=1,JMAX
      DO 120 K=2,KM
      PYR=P(K,J)-P(K-1,J)
      PYP=P(K+1,J)-P(K,J)
      YPR=PYP*PYR
      IF(YPR .LT. 0.0) DPDY(K,J)=0.5*(DPDY(K+1,J)-DPDY(K-1,J))
C     IF(YPR .LT. 0.0) DPDY(K,J)=0.5*(PYP-PYR)
  120 CONTINUE
C
C     CALAULATE THE TRANSFOMATION OF PRESSURE GRADIENTS
C     FORM THE WEIGHT FUNCTION WITH THE FIRST ORDER OF PRESSURE
C     GRADIENT
C
      DO 130 K=1,KMAX
      DO 130 J=1,JMAX
      PXD=DABS(DPDX(K,J))
      PED=DABS(DPDY(K,J))
      PQX=Q(K,6,J)*(PXD*YEF(J,K)-PED*YXF(J,K))
      PQY=Q(K,6,J)*(PED*XXF(J,K)-PXD*XEF(J,K))
      W(K,J)=DABS(PQX)+DABS(PQY)
      W(K,J)=AFA+BTA*W(K,J)
  130 CONTINUE
C
C     CALCULATE THE WEIGHTED FUNCTION SMOOTHLY
C
      DO 160 I=1,IMAX
      DO 140 K=1,KMAX,KM
      DO 140 J=2,JM
      WW=0.5*(W(K,J+1)+W(K,J-1))
      W(K,J)=W(K,J)+0.4*(WW-W(K,J))
  140 CONTINUE
C     W(K,1)=2.*W(K,2)-W(K,3)
C     W(K,JMAX)=2.*W(K,JM)-W(K,JM-1)
      DO 150 J=1,JMAX,JM
      DO 150 K=2,KM
      WW=0.5*(W(K+1,J)+W(K-1,J))
      W(K,J)=W(K,J)+0.4*(WW-W(K,J))
```

```
  150 CONTINUE
C     W(1,J)=2.*W(2,J)-W(3,J)
C     W(KMAX,J)=2.*W(KM,J)-W(KM-1,J)
      DO 160 K=2,KM
      DO 160 J=1,JM
      WW=0.25*(W(K,J+1)+W(K,J-1)+W(K+1,J)+W(K-1,J))
      W(K,J)=W(K,J)+0.4*(WW-W(K,J))
  160 CONTINUE
C
C     CALCULATE THE DERIVATIVES OF WEIGHT FUNCTION
C
      DO 170 K=2,KM
      DO 170 J=2,JM
      WED=0.5*(W(K+1,J)-W(K-1,J))
      WXD=0.5*(W(K,J+1)-W(K,J-1))
      DWX(K,J)= WXD*YEF(J,K)-WED*YXF(J,K)
      DWY(K,J)= WED*XXF(J,K)-WXD*XEF(J,K)
  170 CONTINUE
      RETURN
      END
```

REFERENCES

1. Thompson, J.F., Thames, F.C., and Mastin, C.W. (1974). "Automatic numerical generation of body-fitted curvilinear coordinate system for field containing any number of arbitrary two-dimensional bodies," J. Comp. Phys., vol. 15, pp. 299-319.

2. Mastin, C.W. (1982). "Error induced by coordinate systems," in J.F. Thompson (editor), Numerical Grid Generation, North-Holland, New York, pp. 31-40.

3. Hsu, C.C., and Chang, T.H. (1982). "On a numerical solution of incompressible turbulent boundary layer flow," in T. Kawai (editor), Finite Element Flow Analysis, Univ. of Tokyo Press, Tokyo, pp. 219-226.

4. Thompson, J.F. (editor) (1982). Numerical Grid Generation, North-Holland, New York.

5. Babuska, I., Chandra, J., and Flaherty, J.E. (editors) (1983). Adaptive Computational Methods for Partial Differential Equation, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, Pennsylvania.

6. AIAA Computational Fluid Dynamics Conference. (1983). A collection of technical papers, Danvers, Massachusetts.

7. Brackbill, J.U. (1982). "Coordinate system control: Adaptive meshes," in J.F. Thompson (editor), Numerical Grid Generation, North-Holland, New York, pp. 277-294.

8. Saltzman, J., and Brackbill, J.U. (1982). "Application and generalization of variational methods for generating adaptive meshes," in J.F. Thompson (editor), Numerical Grid Generation, North-Holland, New York, pp. 865-884.

9. Saltzman, J. (1981). "A Variational Method for Generating Multidimensional Grids," Ph.D. Thesis, New York University, New York.

10. Steger, J.L. (1977). "Implicit finite difference simulation of flow about arbitrary geometrics with application to airfoils," AIAA 77-665, Presented at

AIAA 12th Thermophysics Conference at Albuquerque, New Mexico.

11. Pulliam, T.H., and Steger, J.L. (1980). "Implicit finite-difference simulations of three-dimensional compressible flow," AIAA Journal, vol. 18, no. 2, pp. 167-169.

12. Warming, R.F., and Beam, R. (1978). "On the construction and application of implicit factored schemes for conservation laws," SIAM-AMS Proceedings, vol. 2, pp. 85-99.

13. Degani, D., and Steger, J.L. (1983). "Comparison between Navier-Stokes and thin-layer computations for separated supersonic flow," AIAA Journal, vol. 21, no. 11, pp. 1604-1606.

14. Nietubicz, C.J., Pulliam, T.H., and Steger, J.L. (1979). "Numerical solutions of the azimuthal invariant thin layer Navier-Stokes equations," AIAA 79-0010, Presented at AIAA 17th Aerospace Sciences Meeting, New Orleans, Louisiana.

15. Nietubicz, C.J. (1981). "Navier-Stokes computations for conventional and hollow projectile shapes at transonic velocities," AIAA 81-1962, Presented at AIAA 14th Fluid and Plasma Dynamics Conference, Palo Alto, California.

16. Steger, J.L., Nietubicz, C.J., and Heavey, K.R. (1981). "A General Curvilinear Grid Generation Program for Projectile Configurations," Memorandum Report ARBRL-MR-03142. U.S. Army BRL, Aberdeen Proving Ground, Maryland.

17. Sturek, W.B. (1983). "Opportunities for application of adaptive grids in computational aerodynamics," in I. Babuska, J. Chardra, and J.E. Flaherty (editors), Adaptive Computational Methods for Partial Differential Equation, SIAM, Philadelphia, Pennsylvania, p. 185.

18. Kitchens, C.W., Jr., and Mark, A. (1983). "Role of adaptive grid techniques in blast dynamics," in I. Babuska, J. Chardra, and J.E. Flaherty (editors), Adaptive Computational Methods for Partial Differential Equation, SIAM, Philadelphia, Pennsylvania, p. 193.

19. Zienkiewicz, O.C. (1977). The Finite Element Method, Third Edition, McGraw-Hill, New York, pp. 83-85.

20. Winslow, A. (1966). "Numerical solution of the quasilinear poisson equation in a nonuniform triangle mesh," J. Comp. Phys., vol. 149, pp. 153-172.

21. Schlichting, H. (editor) (1978). Boundary Layer Theory, 7th edition, McGraw-Hill, New York, p. 328.

22. Anderson, D.A., Tannehill, J.C., and Pletcher, R.H. (editor) (1984). Computational Fluid Mechanics and Heat Transfer, McGraw-Hill, New York, p. 421.

23. Longley, H.J. (1960). "Methods of Differencing in Eulerian Hydrodynamics," LASL Report LAMS-2379, Los Alamos Scientific Lab, Los Alamos, New Mexico.

24. Gary, J. (1964). "On certain finite difference schemes for hyperbolic systems," Math. of Computation, vol. 18, pp. 1-18.

25. Thompson, J.F., Thames, F.C., and Mastin, C.W. (1977). "Boundary-fed curvilinear systems for solution of partial differential equations on fields containing any number of arbitrary two-dimensional bodies," NASA Contractor Report Cr-2729, Mississippi State Univ., Mississippi State.

26. Viviand, H. (1974). "Conservative forms of gas dynamic equations," La Recherche Aerospatiale 1974-1, pp. 65-68.

27. Vinokur, M. (1974). "Conservation equations of gas-dynamics in curvilinear coordinate system," J. Comp. Phys., vol. 14, pp. 105-125.

28. Baldwin, B.S., and Lomax, H. (1978). "Thin layer approximation and algebraic model for separated turbulent flows," AIAA 78-257, Presented at AIAA 16th Aerospace Sciences Meeting, Huntsville, Alabama.

29. Hsu, C.C. (1983). "Grid Network Generation with Adaptive Boundary Points for Projectiles at Transonic Speed," Technical Report ARBRL-TR-02485. U.S. Army BRL, Aberdeen Proving Ground, Maryland.

30. Kayser, L.D., and Whiton, F. (1982). "Surface Pressure Measurements on a Boattailed Projectile Shape at Transonic Speeds," Memorandum Report ARBRL-MR-03161, U.S. Army BRL, Aberdeen Proving Ground, Maryland.

31. Brackbill, J.U., and Saltzman, J.S. (1982). "Adaptive zoning for singular problems in two dimensions," J. Comp. Phys., vol. 46, pp. 342-368.

BIOGRAPHICAL SKETCH

Chyuan-Gen Tu was born October 10, 1939, in Kiang-Hsi, China. He received a bachelor's degree in June, 1969, in civil engineering from the Cheng-Kung University in Taiwan, Republic of China, and a master's degree in December, 1972, in civil engineering from Colorado State University.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Chen Chi Hsu

Chen-Chi Hsu, Chairman
Professor of Engineering
Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Ulrich H. Kurzweg

Ulrich H. Kurzweg
Professor of Engineering
Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Lawrence E. Malvern

Lawrence E. Malvern
Professor of Engineering
Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Knox T. Millsaps

Knox T. Millsaps
Professor of Engineering
Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Arun Kumar Varma

Arun K. Varma
Professor of Mathematics

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School, and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

May, 1985

Hubert A. Benci

Dean, College of Engineering

Dean for Graduate Studies and Research